

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Strojnícka fakulta

Evidenčné číslo: SjF-5226-87733

**TugShield: miniatúrny prístroj na riadenie
statickej deformácie nosníka**

Diplomová práca

2021

Bc. Eva Vargová

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Strojnícka fakulta

Evidenčné číslo: SjF-5226-87733

TugShield: miniatúrny prístroj na riadenie statickej deformácie nosníka

Diplomová práca

Študijný program: automatizácia a informatizácia strojov a procesov

Študijný odbor: kybernetika

Školiace pracovisko: Ústav automatizácie, merania a aplikovanej informatiky

Vedúci záverečnej práce: prof. Ing. Gergely Takács, PhD.

Konzultant: Ing. Erik Mikuláš

Bratislava 2021

Bc. Eva Vargová



ZADANIE DIPLOMOVEJ PRÁCE

Študentka: **Bc. Eva Vargová**
ID študenta: 87733
Študijný program: automatizácia a informatizácia strojov a procesov
Študijný odbor: kybernetika
Vedúci práce: prof. Ing. Gergely Takács, PhD.
Konzultant: Ing. Erik Mikuláš
Miesto vypracovania: ÚAMAI SjF STU v Bratislave

Názov práce: **TugShield: miniatúrny prístroj na riadenie statickej deformácie nosníka**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Úlohou študenta je navrhnuť a vyrobiť miniaturizovaný experimentálny modul na riadenie statickej deformácie nosníka. Prístroj bude slúžiť na výučbu technických disciplín v automatizácii ako je mechatronika, teória riadenia, identifikácia sústav a spracovanie signálov. Prístroj bude kompatibilný s elektronickým rozložením mikroradičovej prototypizačnej dosky Arduino R3. Študent vytvorí programátorské rozhranie v jazyku C/C++ pre prostriedie Arduino IDE, pre MATLAB, pre Simulink, prípadne iné programovacie jazyky a vývojové prostredia. Ďalšou úlohou je matematicko-fyzikálna analýza dynamického procesu a následná experimentálna identifikácia modelu. Študent taktiež vytvorí rôzne didaktické inštruktážne príklady na spätnovázobné riadenie tohto systému: PID riadenie, lineárno-kvadratické riadenie (LQ), prediktívne riadenie (angl. model predictive control, MPC), prípadne iné metódy riadenia.

V rámci diplomovej práce študent musí

- navrhnuť experimentálne zariadenie, vybrať vhodné elektronické a mechanické komponenty, navrhnuť elektrické zapojenie a dosku plošných spojov, navrhnuť mechanické osadenie, vyrobiť a otestovať funkčnosť navrhnutého modulu;
- napísať programátorské rozhranie (angl. application programming interface, API) na ovládanie zariadenia pre rôzne programovacie jazyky a vývojové prostredia (C/C++ pre Arduino IDE, MATLAB, Simulink, príp. iné);
- opísat dynamický jav matematicko-fyzikálnou analýzou, navrhnuť vhodný testovací signál a na základe meraných výsledkov identifikovať neznáme parametre modelu;
- vytvoriť didaktické úlohy spätnovázobného riadenia metódami PID, LQ a MPC riadenia pre každé vytvorené prostredie.
- využiť prostredie GitHub na manažovanie verzií softvéru a na integráciu do základnej knižnice.

Rozsah práce: 50-70 s.

Riešenie zadania práce od: 15. 02. 2021

Dátum odovzdania práce: 28. 05. 2021

Bc. Eva Vargová
študentka

prof. Ing. Cyril Belavý, CSc.
vedúci pracoviska

prof. Ing. Cyril Belavý, CSc.
garant študijného programu

Čestné prehlásenie

Vyhlasujem, že ja dolepodpisaná som záverečnú prácu vypracovala samostatne pod vedením vedúceho záverečnej práce a s použitím uvedenej literatúry, ktorú som korektnie odcitovala a neporušila som autorské práva tretích osôb.

Bratislava, 30. mája 2021

.....
Vlastnoručný podpis

Moja najväčšia vďaka patrí prof. Ing. Gergelyovi Takacsovi, PhD., za výborné vedenie celej práce. Pod jeho vedením som sa naučila množstvo vedomostí, ktoré využívam v pracovnom prostredí a hlavne samostatnosti a spoľahlivosť sa na vlastný úsudok pri riešení problémov. V druhom rade by som chcela poďakovať konzultantovi, Ing. Erikovi Mikulášovi, ktorý mi pomohol v fažkých časoch nahradíť laboratórium doma. Ďalej by som sa chcela poďakovať Bc. Jakubovi Mihalíkovi, ktorý mi priniesol nový pohľad na vec a bol inšpiráciou pri návrhu funkčného prototypu. V neposlednom rade by som sa rada poďakovala mojim rodičom za to, že mi vytvorili vhodné podmienky pre absolvovanie celého štúdia a uskutočnenie tejto práce. A na záver môjmu životnému partnerovi za to, že mi je takou podporou hlavne v stresujúcich chvíľach ako je napríklad záverečná práca :D.

Bratislava, 30. mája 2021

Bc. Eva Vargová

Názov práce: TugShield: miniatúrny prístroj na riadenie statickej deformácie nosníka
Kľúčové slová: Arduino, TugShield, riadenie, statická deformácia, servomotor,

Abstrakt: V úvode diplomovej práce sa hodnotí súčasný stav techniky výučby automatizácie v praxi a požiadavky kladené na absolventov strojárskeho odboru. Preskúmané sú rôzne projekty, z ktorých každý využíva platformu Arduino ako nízkonákladový, ale pomerne vysoko výkonný hardvér s otvoreným rozhraním Arduino IDE. Ústrednou tému práce je návrh a implementácia nízkonákladovej edukačnej pomôcky na výučbu riadenia, ktorá demonštruje statickú deformáciu nosníka. Zariadenie je založené na senzore ohybu a servomotore, ktorý využíva štandardizované rozhranie Arduina. Druhá kapitola popisuje predchádzajúcu prácu autora, konkrétnie bakalársku prácu, ktorá tvorí základ tejto práce. Tretia kapitola ďalej navrhuje nový a vylepšený hardvér, ktorého cieľom je odstrániť niekoľko obmedzení z predchádzajúcej verzie. Štvrtá kapitola je venovaná vývoju softvéru, vrátane takzvanej knižnice TugShield, so základnými metódami prepojenia hardvéru. Posledná kapitola navrhuje rámec identifikácie systému teoretickým aj experimentálnym spôsobom. Na záver práca poskytuje demonštračné príklady ukazujúce fungovanie a ovládanie TugShieldu.

Title: TugShield: a miniature device for controlling the static deformation of a beam

Keywords: Arduino, TugShield, control, static deformation, servomotor,

Abstract: The introduction to this masters' thesis evaluates the current state-of-the-art in practical automation education and the demands placed on graduates of this field of engineering. Various projects are reviewed presented, each using the Arduino platform as a low-cost but relatively high-performance hardware with the open-source Arduino IDE interface. The central topic of the thesis is the design and implementation of a low-cost control engineering trainer, demonstrating the static deformation of a beam. The device is based on a flexure sensor and a servo motor, utilizing the standardized interface of the Arduino ecosystem. The second chapter describes the previous work of the author, namely the bachelors' thesis that forms the basis of this work. Next, the third chapter proposes a new and improved hardware that aims to remove several limitations from the previous version. The fourth chapter is devoted to software development, including the so-called TugShield library with fundamental methods for interfacing the hardware. The last chapter proposes a system identification framework both in a theoretical and experimental way. Finally, the thesis provides demonstration examples showing the operation and control of the TugShield.

Obsah

Zoznam obrázkov	1
Zoznam premenných a veličín	3
Zoznam skratiek	4
Úvod	5
1 Praktická výučba automatizácie	6
1.1 Situácia v súčasnosti	6
1.2 Automation Shield	7
1.3 Práce na ÚAMAI	8
1.4 Stav praktickej výučby automatizácie vo svete	9
2 TugShield – prvá verzia	13
2.1 Hardvér TugShield_R1	13
2.1.1 Prvý prototyp	13
2.1.2 Druhý prototyp	14
2.1.3 Tretí prototyp	15
2.1.4 Finálny návrh TugShieldu	16
2.2 Softvér TugShield_R1	18
2.2.1 TugShield trieda	18
2.2.2 Demonštračné príklady	20
3 Vývoj nového hardvéru	23
3.1 Modelovanie v LTSpice	25
3.2 Riešenie problému vstupného napäťia	27
3.3 Návrh obvodu podľa základných princípov operačných zosilňovačov	29
3.4 Tvorenie PCB dosky v programe DipTrace	30
4 Programátorské rozhranie pre TugShield_R2	34
4.1 Arduino IDE	34
4.1.1 Knižnica TugShield	34
4.1.2 Metódy	36
4.2 MATLAB	37
4.3 Simulink	40

5 Demonstračné príklady	45
5.1 Matematicko-fyzikálna analýza	45
5.1.1 Experimentálna analýza	48
5.2 Príklady	49
5.2.1 Príklad FirstCheck	50
5.2.2 Príklad SelfTest	50
5.2.3 Príklad Identification	52
5.2.4 Príklad s použitím PID regulátora	54
6 Záver	57
6.1 Priestor na zlepšenie	58
Literatúra	59

Zoznam obrázkov

1.1	Dopyt po pracovných odvetviach [31].	6
1.2	MotoShield [37] a MagnetoShield [22].	8
2.1	Prvý prototyp [38].	14
2.2	Tretí prototyp [38].	15
2.3	Schéma zapojenia [38].	16
2.4	Predná a zadná strana finálnej PCB dosky [38].	17
2.5	Finálny prototyp [38].	18
2.6	Výstup z príkladu identifikácie systému [38].	21
2.7	Výstup z príkladu PID [38].	22
3.1	Diferenčné zapojenie operačného zosilňovača.	24
3.2	Diferenčné zapojenie s ideálnymi zdrojmi.	25
3.3	Diferenčné zapojenie s reálnym signálom z flexi snímača.	26
3.4	Aplikácia LM317.	27
3.5	Schéma zapojenia s 3 operačnými zosilňovačmi.	28
3.6	Model ideálneho operačného zosilňovača.	29
3.7	Ideálny operačný zosilňovač s dvomi rezistormi.	30
3.8	Simulácia návrhu obvodu s jednoduchou logikou.	30
3.9	Schéma zapojenia TugShield_R2A.	31
3.10	Schéma zapojenia TugShield_R2B.	32
3.11	PCB dosky R2A a R2B.	32
3.12	Hotový TugShield R2B.	33
4.1	Reťazec blokov vytvárajúci ActuatorWrite.	41
4.2	Blok ActuatorWrite.	41
4.3	Maska bloku ActuatorWrite.	41
4.4	Interaktívna maska bloku SensorRead	42
4.5	Blok SensorRead.	42
4.6	Vnútro bloku SensorRead.	42
4.7	TugShield blok.	43
4.8	Spojenie blokov ActuatorWrite a SensorRead vo vnútri bloku	43
4.9	Simulink knižnica AutomationShield so zahrnutým TugShieldom	44
5.1	Závislosť výstupného signálu od uhla ramena serva.	46
5.2	Závislosť ohnutia nosníka od uhla ramena serva.	47
5.3	Výsledky pre TugShield_R2A a TugShield_R2B z príkladu FirstCheck.	50

5.4	Výsledom príkladu SelfTest v Matlabe.	52
5.5	Výstup príkladu Identification v Arduino IDE.	53
5.6	Návrh štruktúry príkladu Identification v prostredí Simulink.	54
5.7	Návrh štruktúry príkladu PID v prostredí Simulink.	56

Zoznam fyzikálnych veličín

Symbol	Veličina	Jednotka (iná jednotka)
P	výkon	W (MW)
A_u	zosilnenie	V/V
C	kapacita	μF
U	napätie	V
U_{in}	vstupné napätie	V
U_{out}	výstupné napätie	V
I	prúd	A
F	sila pôsobiaca na nosník	N
I	elektrický prúd	A (mA)
V_{out}	vstupné napätie	U
M	vnútorný ohybový moment	Nm
E	Youngov modul	Pa
J_z	plošný moment zotrvačnosti prierezu	mm^4
w	uhol vychýlenia	$^\circ$
q	rozložené zaťaženie	$\text{N}\cdot\text{m}^{-1}$
t	čas	s (ms)
L	dĺžka	m (mm)
R	elektrický odpor	Ω
U	elektrické napätie	V
δ	pružné vychýlenie	$^\circ$
θ	uhol vychýlenia	rad

Zoznam skratiek

Skratka	Význam
3D	trojrozmerný
AC	striedavý prúd
ADC	analógovo-digitálny prevodník
API	rozhranie pre programovanie aplikácií
CAD	počítačom podporovaný návrh
CNC	počítačom riadený stroj
DAC	digitálno-analógový prevodník
DC	jednosmerný prúd
GND	uzemnenie
IDE	integrované vývojové prostredie
LDR	rezistor závislý od svetla
LED	luminiscenčná dióda
LQ	lineárny kvadratický regulátor
MCU	mikrokontrolér
MPC	prediktívne riadenie na základe modelu
PCB	doska plošných spojov
PD	proporcionálno-derivačný regulátor
PID	proporcionálno-integračno-derivačný regulátor
PWM	modulácia periodického signálu
USB	univerzálna sériová zbernice

Úvod

Momentálny stav na akademickej pôde v technických odvetviach nie je zrovna najľahší. Univerzity nemajú dostatok zdrojov na zabezpečenie odborného vybavenia pre všetkých študentov a preto študenti vykonávajú experimenty len v obmedzených množstvách a pod dozorom vyučujúcich.

Situácia sa ešte výrazne zkomplikovala, keď nastala pandémia a štúdium sa presunulo do virtuálneho sveta. Pre študentov automatizácie, ktorí prahnu po nových vedomostiah i po praktických skúsenostiach, to je náročné, aj keď sa univerzity snažia spraviť maximum pre pripravenosť svojich absolventov.

Základom automatizácie je riadenie, ktoré má špecifickú formu výuky, ktorá prechádza od teórie po riadenie reálnych systémov v praxi. Pre tento účel sa v rámci projektu AutomationShield, navrhlo miniatúrne zariadenie na riadenie statickej deformácie nosníka s využitím platformy Arduino s názvom TugShield.

Úvodom diplomovej práce sú uvedené rôzne iniciatívy univerzít po celom svete, ktoré si amatérsky vytvárajú pomôcky na výučbu s implementáciou riadenia. Vo viacerých prípadoch je použité práve Arduino, ktoré je cenovo dostupné. Nevýhodou projektov je však takmer žiadna dokumentácia a s tým súvisí minimálna reprodukovateľnosť.

Druhá kapitola sa venuje predošej bakalárskej práci, na ktorú diplomová práca nadvázuje. Popisuje sa prvý prototyp TugShieldu, jeho podrobny vývoj, najmä hardvérovej časti. Zvýraznené sú nedostatky tejto práce a možnosti zlepšenia, ktoré sú odrazovým mostíkom aktuálnej záverečnej práce.

V tretej kapitole je kompletne a podrobne opísaný vývoj nového prototypu, ktorý prešiel niekoľkými fázami. Rieši sa tu otázka zmeny typu zapojenia operačného zosilňovača, poradie komponentov v elektrickom obvode či umiestnenie na prototypizačnú dosku.

Softvérové rozhranie pre užívateľa je tému štvrtnej kapitoly. Navrhujú sa API pre tri prostredia a to: Arduino IDE, MATLAB a Simulink, ktoré každé z nich prináša rôzne úskalia. Základom všetkých rozhraní je knižnica TugShield, ktorá obsahuje metódy a funkcie pre ovládanie základných činností TugShieldu.

Pre riadenie bola načrtnutá analýza. Teoreticky aj experimentálne sa navrhlo získavanie modelu pre tento systém, kde jadro bola štrukturálna deformácia nosníka. V piatej kapitole sa spolu s analýzou navrhli aj demonštračné príklady, ktoré majú za úlohu priblížiť užívateľovi fungovanie TugShieldu a spolu s tým sa overuje aj správnosť vytvorených metód.

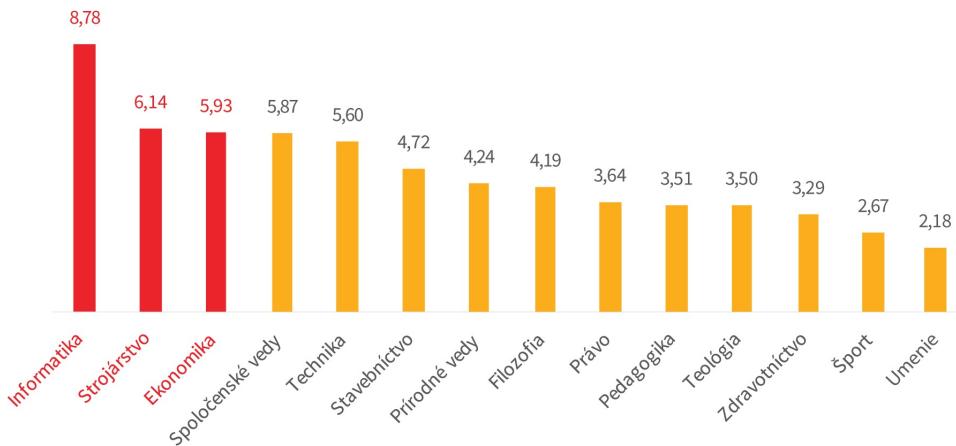
1 Praktická výučba automatizácie

1.1 Situácia v súčasnosti

Situácia pre absolventov vysokých škôl na trhu práce je momentálne mimoriadne náročná. Z dôvodu krízy mnoho podnikov muselo redukovať svoje stavby, obmedziť výrobu, zrušiť časti prevádzok či úplne zanikli. Samozrejme to malo veľký vplyv na trh práce [41]. Dopyt po nových zamestnancov klesol a tak isto aj dopyt po čerstvých absolventoch.

Na končiacich študentov je vyvíjaný tlak v oblasti presadenia sa na trhu práce. Požiadavky sa zvyšujú a stále sa rozširuje škála zručností a znalostí, ktoré by mal študent ovládať [3]. Koncom minulého storočia bola veľká výhoda absolventa znalosť jazykov, čo sa dnes považuje za samozrejlosť a za dve desaťročia sa k samozrejmostiam pridala aj práca s počítačom.

Pokrok v oblasti technológií sa podpísal na riadení a manažovaní firiem, kde správny hardvér v spolupráci so softvérom vie uľahčiť a zjednodušiť procesy a čiastočne aj nahradieť pracovnú silu.



Obr. 1.1: Dopyt po pracovných odvetviach [31].

Na Slovensku prevláda priemyselná výroba. Sme svetová dominanta v automobilovom priemysle a preto aj najviac pracovných ponúk je vytvorených práve v tejto oblasti [32]. Núdza je o pracovníkov so zameraním na strojársky odbor a tak isto aj na informačné technológie. Príjemným spojením týchto dvoch odvetví je práve automatizácia, ktorá sa vo veľkej časti zaobrásť riadením procesov a strojov s minimálnym zásahom človeka, teda s využitím počítačových technológií.

Aj napriek tomu, že sú študenti automatizácie žiadani, kladú sa nich vysoké požiadavky. Veľkú výhodu majú študenti, ktorí mali možnosť zapojiť sa do pracovného procesu už počas vysokej školy a nadobudnúť tým praktické zručnosti a vedomosti, ktoré sa na akademickej pôde získavajú nie ľahko. Univerzity a jednotlivé fakulty či ústavy situáciu dobre poznajú a preto sa snažia študentom čo najviac vyhovieť a umožniť im prácu v špecializovaných laboratóriách, kde tieto zručnosti z praxe môžu alternatívne nadobudnúť.

Pre vysoké školy to však vôbec nie je jednoduchá úloha, ako sa môže zdať. Mnohokrát musia čeliť nedostatkom personálu a ešte väčším nedostatkom laboratórneho vybavenia. Študenti nemajú povolené zostať v laboratóriách sami. Ich práca musí byť kontrolovaná, jednak kvôli bezpečnosti, ale aj nástroje s ktorými pracujú stoja desiatky tisíc eur. Laboratória sú obvykle zastaralé a študenti tak neprídu do kontaktu s najnovšími technológiami, ktoré sú v praxi zaužívané.

Ako sme si aj v poslednom období mohli overiť, práca z domu je niekedy nevyhnutná. Pre väčšinu je to príjemná zmena, ale len s ťažkosťami sa dá nahradí doma laboratórne prostredie. Univerzity nedisponujú dostatkom odborného a technického vybavenia na to, aby ho mohli zapožičiavať študentom na prenesenie svojich experimentov domov prípadne na internát, kde na nich môžu pracovať 24/7.

1.2 Automation Shield

V roku 2018 vznikla na ústave ÚAMAI iniciatíva s názvom AutomationShield [14], ktorá sa snaží riešiť práve vyššie uvedené problémy. Jej hlavným cieľom je vytvárať edukačné pomôcky pre študentov vysokých škôl, ktoré sú zamerané na automatizáciu. Ako už z názvu vyplýva, jedná sa o dodatočný hardvér na platformu Arduino Uno takzvané "shieldy".

Dôvody na použitie práve Arduina Una sú v prvom rade Open-source zdrojový kód, nízke cenové náklady a programovacie prostredie Adruino IDE, ktoré kombinuje jazyky C a C++. Mikroprocesorová doska na báze Atmel ATmega328, má tiež 8-bitový vysoko výkonný AVR mikrokontrolér [5]. Ďalšou, nie malou výhodou je, že Arduino Uno má hardvérovú verziu rozdelenia pinov R3, ktorá je využívaná viacerými výrobcami mikroprocesorových dosiek, teda vyrobené shieldy sú kompatibilné aj s doskami ako napríklad Genuino Uno, Adafruit Metro express, M4 express a ďalšie.

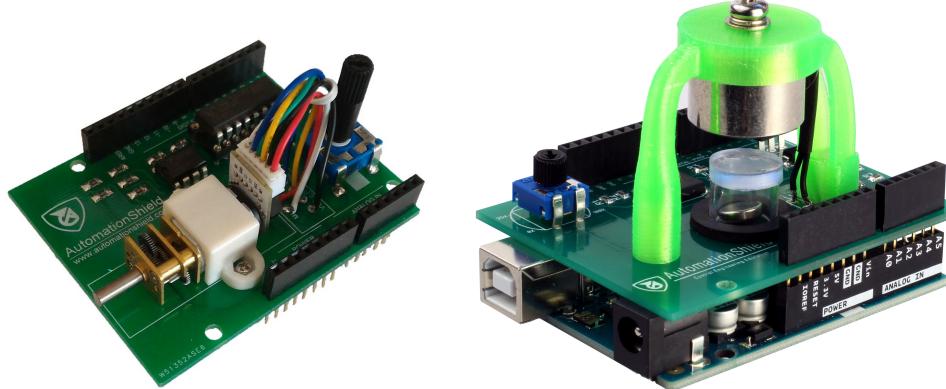
Základ hardvéru AutomationShield tvorí vždy prototypizačná doska, ktorá je prispôsobená na jednoduché zasunutie do pinov Arduina. Na doske sú ďalej umiestnené špecifické komponenty, ktoré sú pre každý projekt iné a spolu s doskou tak vytvárajú pomôcku pre výuku spätno-väzobného riadenia.

Zahrnutie vzdelávacích pomôčok Automation shield do výučby má viaceré výhody. V prvom rade je to cena shieldov. Celkové náklady na zhotovenie sa pohybujú od 2–12€závisí podľa projektu. Samotné vyhotovenie je veľmi jednoduché, keďže projekty sú precízne zdokumentované, súčiastky sa dajú jednoducho objednať z internetových predajní a špeciálne komponenty pre 3D tlač sú navrhnuté v CAD programe. Kompletná dokumentácia je voľne dostupná a nachádza sa na stránke [github/automationshield](https://github.com/automationshield).

Ku každému shieldu je vypracovaná aj knižnica pre prostredie Arduino IDE, MATLAB a Simulink. Študenti môžu tieto pomôcky využiť na pochopenie základných princímov programovania, riadenia a v prípade, že si prototyp sami zhotovia, oboznámia sa s fungovaním snímačov, spájkovaním a 3D tlačou.

V tomto projekte vzniklo už niekoľko shieldov, ktorý každý disponuje pred tým spomenutými vlastnosťami, no princíp fungovania je odlišný.

Ako prvý zo série bol vytvorený MotoShield [37]. Jeho základnými komponentmi sú motor s prevodovkou, ktorý vystupuje ako riadiaci člen a enkodér, cez ktorý sa získavajú výstupné dáta. MotoShield sa dá využiť pri pokusoch riadení rýchlosťi či polohy.



Obr. 1.2: MotoShield [37] a MagnetoShield [22].

MagnetoShield je založený na báze elektromagnetických síl [22]. Elektromagnet generuje magnetickú silu, ktorá dvíha permanentný magnet. Poloha levitujúceho magnetu sa zistuje pomocou snímača, ktorý pracuje na Hallovom efekte a tým sa dá magnet nepriamo regulovať.

Vznášajúci sa predmet je základom aj FloatShieldu, avšak nie za pomocí magnetickej sily, ale prúdiaceho vzduchu [20]. Guľa vyrobená z ľahkého materiálu je umiestnená do vertikálnej trubice, na ktorej jednom konci sa nachádza snímač vzdialenosťi a na druhom ventilátor. Pomocou výkonu ventilátora riadeného cez potenciometer sa docieľuje určitá výška gule v trubici.

Ďalším zo série je OptoShield [23]. Na PCB doske sa nachádza LED dióda, ktorej svietivosť sa môže regulovať buď manuálne pomocou potenciometra alebo sa môžu zadať vstupné hodnoty programovo cez Arduino IDE. V trubici, ktorá eliminuje okolité svetlo sa okrem riadiaceho člena LED diódy nachádza ja LDR rezistor, ktorého odpor je závislý od svetla.

Princípy termoregulácie si študenti môžu osvojiť cez HeatShield [21]. V dostatočnej izolácii od PCB dosky je aplikovaná vyhrievacia kazeta, ktorá ohrieva vnútro komponentu vytlačeného na 3D tlačiarni. Teplota je zachytávaná odporovým snímačom teploty a spolu s kazetou vytvárajú spätno-väzobnú slučku.

Návrh prototypov a vypracovanie celej dokumentácie boli realizované študentami v rámci ich záverečných prác, s pomocou ich konzultantov a vedúcich práce. Samozrejme vyššie spomenuté shieldy nie sú jediné projekty v AutomationShield, ďalšie sú vo vývoji.

1.3 Práce na ÚAMAI

Na Ústave automatizácie, merania a aplikovanej informatiky sa študenti aktívne podieľajú na tvorbe nových učebných pomôcok na automatizáciu. Niektorí sa zameriavajú čisto len

na konštrukciu hardvéru alebo len na vývoj softvéru a iní pracujú na komplexnejších návrhoch, kde výsledkom sú hotové zariadenia.

Cieľom diplomovej práce Bc. Filipa Čelka bolo od základu navrhnúť konštrukciu, ale aj softvér pre experimentálne zariadenie vhodné na aplikáciu adaptívneho prediktívneho riadenia [39]. Hlavné dva komponenty tvorili motor a brzda. Zariadenie malo byť schopné pracovať v dvoch režimoch, v prvom manuálnom a druhom automatickom riadenom cez prostredie Matlab/Simulink. Autor projektu riešil aj otázky merania otáčok, pričom porovnával viaceré metódy merania. Na vytvorené zariadenie aplikoval jednoduché spätnoväzbové riadenie - PID regulátor [39].

Motor bol zakomponovaný aj do ďalšej z prác [26]. Išlo o riadenie uzatvárania škrtiacej klapky na motore. Klapka slúži na riadenie prietoku vzduchu a má vplyv na otáčky a rýchlosť motora. Riadenie sa ukázalo ako zaujímavý experiment z dôvodu nonlinearity pružiny na klapke a potrebe vysokej rýchlosťi riadenia. Autor zostavil matematický model a PWA model, na ktoré aplikoval nie len PID regulátor, ale aj metódu prediktívneho riadenia.

Projekt, ktorý sa čisto venoval iba softvérovému návrhu, bol projekt Bc. Jakuba Kulháneka, BSBA. Práca sa zaoberala implementáciou riadiacich algoritmov na laboratórnom modeli obráteného kyvadla [7]. Na navrhovanom systéme sa sformuloval matematický model a rozšírený Kalmanov filter, ktorý sa použil ako pozorovateľ stavu. Študent na riadenie nevyužil platformu Arduino Uno, ale dosku STM32F4 Discovery vybavenú mikrokontrolérom STM32F407VG.

1.4 Stav praktickej výučby automatizácie vo svete

Ústav automatizácie a informatizácie v Bratislave nie je ale jediným, ktorý si vytvára pomôcky svojpomocne či zapojil Arduino do výučby automatizácie a spätno-väzobného riadenia.

Projekt "Lab-In-A-Box" Standfordskej univerzity na katedre elektrotechniky zlepšuje výučbu spracovania signálu a analógovej elektroniky. Realizuje sa pomocou dvoch dosiek spolu so sprievodným softvérovým balíkom. Prvá doska – Analógový Shield – je digitálno-analógový (DAC) a analógovo digitálny prevodník (ADC), ktorý sa používa v niekoľkých príkladoch, ktoré jednoducho vysvetľujú koncept signálov v praxi. Druhá z dosiek – DSP shield – spája procesor digitálneho signálu s Arduinom a umožňuje tak rôzne možnosti zložitosti pri výučbe signálových aplikácií, od implementácie algoritmu na nízkej úrovni až po vysokú [12].

V roku 2018 skonštruoval Dimitrios Iosifidis a spol. [16] v rámci svojej diplomovej práce laboratórny dron. Tento projekt mal za úlohu ovládať vznášania sa kvadrokoptéry vyrobenej na zákazku s obmedzeným pohybom. Dron bol modelovaný len na pohyb v jednej osi a umožňoval tak bezpečnú prevádzku v laboratórnom prostredí, pričom ponecháva dostatočný stupeň voľnosti pre užitočné experimenty s PID riadením, ktoré sú podľa názoru vedúceho projektu často ignorované vo vzdelávacích osnovách. Celkové náklady na vývoj tohto zariadenia sa udržali na nízkej úrovni vďaka použitiu moderného lacného hardvéru a softvéru založeného na platforme Arduino.

Na Nigerian Turkish Nile University už niekoľko rokov aktívne využívajú Arduino na spestrenie štúdia. Uvádzajú sedem dôvodov, prečo používať Arduino ako nástroj na učenie

[13]:

- aktívna komunita užívateľov,
- platforma vyvinutá vo vzdelávacom prostredí,
- cenovo dostupný hardvér,
- vývojové prostredie vhodné aj pre amatérov,
- programovateľné pomocou USB kábla,
- open-source hardvér aj softvér,
- multiplatformové vývojové prostredie (Microsoft, Linux a Mac OS X).

Dôvody poukazujú na výhody ako pomoc v prípade potreby od iných užívateľov, nízke náklady, blízkosť platformy Arduino k študentom, jednoduchosť používania a vhodnosť pre programátorov - začiatočníkov. Študenti Nigérijskej univerzity pracujú na projektoch elektronické textílie, záznamník údajov ORP / pH / teplota a Arduino Satelit [13]. Využívajú už navrhnutý hardvér a aplikujú ho na vlastné kreatívne nápady.

V susednom Česku v Českých Budějoviciach rozbehli projekt s názvom PRIM [28], ktorého hlavná úloha je podpora rozvoja informatického myslenia a logiky. Výstup projektu sú súbory metodík, materiálov a príkladov použiteľných ako pedagogický sprievodca pre vyučujúcich na vysokých, ale aj stredných školách. Učebné osnovy obsahujú návody ako narábať s mikroprocesorovou doskou Arduino a dosiahnuť tým vynikajúce študijné výsledky pri riešení problémov z reálneho života.

Katedra elektrotechniky v Brazílii v spolupráci so strednou školou v Portugalsku sa presvedčili o tom, že praktické experimenty, vykonávané priamo študentmi, prispievajú k nie len spestreniu vyučovania, ale aj k chápaniu učiva [30]. Jedenásť hodín uskutočňovali fyzikálne experimenty z oblasti optiky, termodynamiky a vlnenia žiaci v šesť členných skupinkách. Platforma Arduino tvorila základ hardvéru každej skupiny a zabezpečila tak prístup k technickým nástrojom všetkým študentom. Až 94 % zo zúčastnených študentov ocenilo tento prístup k vyučovaniu fyziky a povedalo, že na rozdiel od obyčajných teoretických hodín, kde s profesionálnym laboratórnym vybavením pracuje len vyučujúci, boli hodiny zaujímavé až veľmi zaujímavé [30]. Zvýšilo to ich motiváciu a z veľkej miery pomohlo lepšie pochopiť učivo, avšak náročnosť a zložitosť predmetu sa nezmenila.

Dopyt po elektrine a elektrifikácii sa zvyšuje s pokrokom technológií, čo spôsobuje v niektorých Afričkých krajinách problémy s pokrytím dopytu. Vo vyspelých mestách je to riešené za pomoci inteligentných elektrických spínačov (IED), ktoré snímajú nadmerné preťaženie siete a odstavia časť distribúcie energie v prípade, že je záťaž väčšia ako generovaná kapacita energie [29]. V dôsledku zlého plánovania a náhodného charakteru rastu miest v chudobnejších krajinách v Afrike, vzniká mnohokrát stav preťaženia siete. IED sú aplikované v nedostačujúcom množstve, prípadne sú zastarané.

Situáciu v štyroch obytných zónach nasimulovali študenti pod vedením Olugbenga K. Ogida [29]. Architektonický návrh využíva Arduino kombinované so spínacím obvodom na implementáciu plánu, ktorý je v súlade s oficiálnou distribučnou spoločnosťou energie v danej oblasti. Model s naprogramovaným mikrokontrolérom je využívaný ako automatický

spínač a tým sa optimalizuje využitie dostupného výkonu 7,5 MW bez ľudského zásahu, pričom spotreba energie je 10 MW. Vytvorený model je užitočný ako pomôcka pri výučbe automatizácie energetických systémov.

Vývoj platformy Arduino sa začal v Taliansku v roku 2005 a odvtedy si získal v komuniti, ktorú už predstavuje viac ako 100 000 aktívnych používateľov, veľkú popularitu [15].

V porovnaní s inými programovateľnými mikrokontrolérmi, Arduino nepotrebuje špeciálny hardvér na načítanie logiky do dosky. Každá doska môže byť pripojená k rôznym iným modulom. Mnoho vysokých škôl zameraných na techniku už zahrnulo Arduine do svojich predmetov a výnimkou nie sú ani univerzity na Slovensku. Z celkového počtu 13 univerzít po celom svete práve 3 univerzity sú na Slovensku a to Žilinská univerzita, Technická univerzita v Košiciach a Slovenská technická univerzita [15].

V súčasnom trende modernizácie vo výrobe, využitie CNC obrábacích strojov rastie exponenciálne. V jadre práce CNC stroja je úloha interpolátora veľmi významná. Projekt Technologickej univerzity v Gujaratе v Indii sa týka návrhu a vývoja riadiacej jednotky pozostávajúcej práve zo softvérového interpolátora ako základného prvku pri riadení pohybu CNC stroja, ktorý plne nahradza pôvodný hardvérový interpolátor [9]. Navrhnutá riadiaca jednotka prispieva malým, ale významným krokom k automatizácii a môže šetriť vysoké náklady spojené s ňou.

V poslednom desaťročí sa svet čoraz viac stretáva s problémom nedostatku kvalitných potravín. Jedným zo spôsobov, ako čiastočne vyriešiť tento problém navrhuje Polytechnická Národná Univerzita, ktorá používa inteligentné skleníkové technológie na pestovanie plodín [36]. Tieto skleníky sa nachádzajú vo vnútri obytných zón a umožňujú rast rastlín po celý rok. Vývoj automatizovaného riadiaceho systému na správu malých skleníkov je založený na mikrokontroléri Arduino a umožňuje riadenie pomocou Androidu. Vyvinuli inteligentný systém na manažovanie skleníka, ktorý je schopný analyzovať stav mikroklímy v skleníku, ale tak isto ho aj ovplyvňovať. Systém je autonómny, schopný regulovať stav klímy a osvetlenia, aby sa zachovali priaznivé podmienky pre rast rastlín. Okrem toho je jednoducho nastaviteľný a dlhodobo udržateľný.

Podobný projekt vytvorili aj v Indii s tým rozdielom, že sa zamerali na maximálny rast rastlín [40]. Prispôsobili podmienky na vystavovanie čo najviac plodín, v čo najkratšom čase, ale na druhej strane to malo dopad na kvalitu.

Asi jeden z najobľúbenejších príkladov použitia Arduina kombináciou s riadením predstavuje regulácia otáčok motora [4], [6], [18]. Využitie motorov v praxi je nespočetné ako napríklad ventilácia, klimatizácia, motory v autách či v domáčich spotrebičoch.

Najčastejšie sa však v laboratóriách stretávame s radením jednosmerných motorov s permanentným magnetom. V Indii aplikovali riadenie DC motora v dvoch experimentoch. V prvom sa zamerali čisto len na riadenie otáčok a smeru motora pomocou Arduina a prostredia LabVIEW [4]. V druhom experimente boli otáčky motora závislé od teploty, keďže príklad bol stavaný ako návrh klimatizácie či ventilátora [6].

Turci sa na motory pozreli z hľadiska bezpečnosti v premávke [18]. Nadmerná rýchlosť je častou príčinou nehôd, zranení, ale aj ničenie verejného majetku a preto vytvorili ultra-zvukový radar, ktorý meria rýchlosť vozidiel. Pokial vodič prekročil maximálnu dovolenú rýchlosť v prvom rade bude upozornený a ak nerešpektuje výstrahu, vozidlo spolu s ŠPZ bude odfotené a poslané príslušným orgánom.

Lasery s pulznou dobou trvania 100 ns sa široko používajú v ablatívnych procesoch.

Vo všeobecnosti existujú rôzne aplikácie týchto laserov ako napríklad v biomedicíne, priemyselných oblastiach, ale aj pri ochrane kultúrneho dedičstva.

V práci [8] vo Florencií integrovali ručný systém z optických vlákien s inovatívnou spätnou väzbou na riadenie laserového toku v ohniskovej oblasti. Zamerali sa predovšetkým na selektívne odstraňovanie nežiadúcich vrstiev materiálu (ako napríklad farba, hliná ...) z predmetov s historickou hodnotou. Laserový tok je závislý od odrazu vzorky laseru od povrchu, čím sa rozozná druh a šírka materiálu vrstvy a následne sa pri odstraňovaní ochráni predmet kultúrneho dedičstva. Takéto laserové ošetrenie je bezpečnejšie, presnejšie a produktívnejšie, pričom sa plne využíva spätno-väzobné riadenie v oblasti v akej by sa to len ľažko dalo čakať.

PID regulátor je klasický trojzložkový regulátor, ktorý sa v priemysle bežne používa a má veľa výhod. V prvom rade je veľmi jednoduchý, a podľa potreby je možné použiť len niektoré zložky z neho. Napríklad PD regulátor zlepšuje tlmenie a znižuje maximálne prekročenie a zase PI regulácia môže znížiť alebo eliminovať chybu v ustálenom stave.

V článku [19] navrhli inovatívny laboratórny študijný program pre vysoké školy s technickým zameraním. Hlavným komponentom je systém SOAC, ktorý najprv testovali len na doske plošných spojov pripojeným na osciloskop. Neskôr systém implementovali na Arduino a výsledkom bola vzdelávacia pomôcka pre lepšie pochopenie PID regulátorov. Z experimentálnych výsledkov si študent môže nadobudnúť praktické skúsenosti a porozumieť riadiacim systémom.

Riadiace inžinierstvo je interdisciplinárne pole, ktoré si vyžaduje znalosti z matematiky, fyziky, elektrických obvodov, snímačov, akčných členov a mikrokontrolérov. Poskytuje tiež skúsenosti používateľovi pri testovaní, simulácii a implementácii v reálnom čase. Typickým systémom na vzdelávanie v tejto oblasti je nádrž na vodu. Dynamika systému spočíva v zmene hladiny vody, ktorá je závislá od regulovaného prítoku a výpustu.

Článok [33] sumarizuje evolúciu projektu vo výučbe na kontrolu a riadenie hladiny vody pre študentov vysokých škôl na UW Tacoma. Projekt predstavuje kontrolu hladiny laboratórnej nádrže na vodu v reálnom čase a demonštruje podstatu kontrolného inžinierstva. Cieľom je navrhnuť a použiť PID regulátor na udržanie hladiny vody v nádrži pomocou Arduina. Po modelovaní systému a testovaní sa navrhne regulátor a simuluje sa systém uzavretej slučky. Vylepšený regulátor sa potom implementuje. Porovnaním simulácie a experimentálnych výkonov systému s uzavretou slučkou sa diskutuje o rozdieloch spôsobených nesúladom medzi realitou a modelom.

2 TugShield – prvá verzia

Dodatočný hardvér a softvér pre platformu Arduino Uno, konkrétnie TugShield prvá verzia, sa začal vyvíjať v roku 2018, ako záverečná práca pre bakalársky stupeň štúdia [38]. Motiváciou pre skonštruovanie tohto zariadenia bolo rozšírenie didaktických pomôcok pre výučbu teórie automatického riadenia, kde by si študenti priamo mohli vyskúšať naprogramovať a vyladiť napríklad PID, LQ alebo MPC regulátor.

TugShield_R1 funguje na jednoduchom princípe spätnovázobnej reakcie. Na jednej strane sa nachádza servomotor, ktorého rameno je spojené s flexi snímačom na druhej strane. Teda cez vstupný signál, ktorý sa zadáva do servomotora, ako uhol je následne ohýbaný flexi snímač, ktorého výstupom je hodnota úmerná ohybu. Celá práca obsahovala dve základné problematiky a to vývoj prototypu hardvéru a vývoj softvéru.

2.1 Hardvér TugShield_R1

Návrh prototypu hardvéru TugShield_R1 bola komplexná úloha zahrňujúca poznatky z viacerých oblastí, ale predovšetkým z elektroniky a konštruovania. Základom bolo navrhnuť shield, ktorý bude mimoriadne praktický, jednoduchý, spratný, odolný voči poškodeniu pri používaní a manipulácií študentami, ale zároveň, aby poskytoval širokú škálu možností využitia. Toto všetko ovplyvňovalo voľbu súčiastok, kde hlavným faktorom bola spratnosť a funkčnosť. Vytvoreniu finálnej verzie TugShieldu_R1 predchádzali tri prototypy, ktoré sa s každou ďalšou verzou vyvíjali.

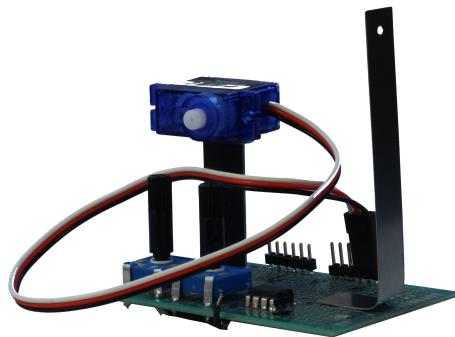
2.1.1 Prvý prototyp

Na prvom prototype sa riešila hlavne otázka celkového uloženia komponentov. Základom každého shieldu bola PCB doska pre Arduino UNO, ktorej rozmery sú $68,58 \times 53,3$ mm. S doskou sa dobre pracovalo, pretože sa na ňu dali jednoducho napájkovať a odpájkovať komponenty v prípade potreby.

Ďalší dôležitý komponent bol snímač ohybnosti – flexi senzor, dlhý celkovo 73,66 mm. Snímač je tenký len 0,43 mm [34] a tým pádom príliš mäkký na to, aby stál bez podpory, musel sa nalepiť na kovový nosník. Nosník bol vyrobený z kalenej ocele a preto sa potrebné rozmery získali lasérovým rezaním. Keďže hrana PCB dosky a flexi senzor merali obidva približne 7 cm, uvažovali sa dva varianty umiestnenia snímača.

Prvé uloženie malo byť vodorovné a to tak, že by sa snímač umiestnil presne na dlhšiu hranu dosky. Treba však spomenúť, že tretím a posledným nosným prvkom v hardvérovej skupine je servomotor, ktorého úloha bola ohýbať senzor ohybu. V prípade, že by sa flexi umiestnilo vodorovne na hranu PCB dosky, na opačnej hrane by sa muselo nachádzať

servo, ktorého rameno bolo spojené s nosníkom a tak aby ho bolo schopné ohýbať. Vzdialenosť medzi týmito dvoma komponentmi by bol len 4 cm a ohyb by bol veľmi obmedzený touto malou vzdialenosťou. Preto bol zvolený druhý variant a to uloženie nosníka s nalepeným flexi snímačom zvislo, čím sa získala vzdialenosť od ramena servomotora o 3 cm väčšia, teda dokopy 7 cm [38].



Obr. 2.1: Prvý prototyp [38].

Nosník mal tvar L, aby sa dal jednoducho prilepiť na PCB dosku kratšou stranou a dlhšia strana bola poskytnutá snímaču ohybu. Na vrchu nosníka bola vytvorená diera, aby sa nosník dal spojiť s ramenom serva šnúrou.

Na prvom prototype bolo servo umiestnené na 4 cm vysokom plastovom stĺpe, aby bolo približne v rovnakej výške ako vrch flexi senzoru. Po testovaní a logickej úvahе sa však prišlo na to, že to nebola dobrá voľba, pretože tým pádom rameno ohýbalo len vrch snímača a spodná časť zostala nevyužitá. Prvý prototyp riešil aj testovanie zostaveného prvého elektrického obvodu. Flexi snímač sa zapájal do obvodu ako delička napäťa spolu s rovnocenným rezistorom. Hodnota tohto rezistora sa musela zistiť experimentálne a rovnako experimentálne sa aj zistilo, že výstupný signál z flexi senzoru nevyužíva celý rozsah ADC úrovni a preto sa musel do obvodu umiestniť aj operačný zosilňovač [38].

Zapojenie opampu, konkrétnie LM358, bolo zvolené neinvertujúce. Je to jedno z najjednoduchších zapojení, ktoré zachováva pôvodnú logiku signálu a platí jednoduchá logika zosilnenia výstupného signálu $A_u = 1 + R_2/R_1$. Na zistenie hodnôt R_1 a R_2 sa miesto rezistorov s permanentnou hodnotou použili na prvý prototyp potenciometre s nastaviteľným odporem. Prvý prototyp bol len veľmi počiatočným a amatérskym návrhom. Keďže neobsahoval kolíky na priame zasunutie do pinov Arduina na spojazdnenie sa museli použiť navyše káble a bezpájkové kontaktné pole.

2.1.2 Druhý prototyp

Druhý návrh TudShieldu_R1 bol už o niečo sofistikovanejším riešením.

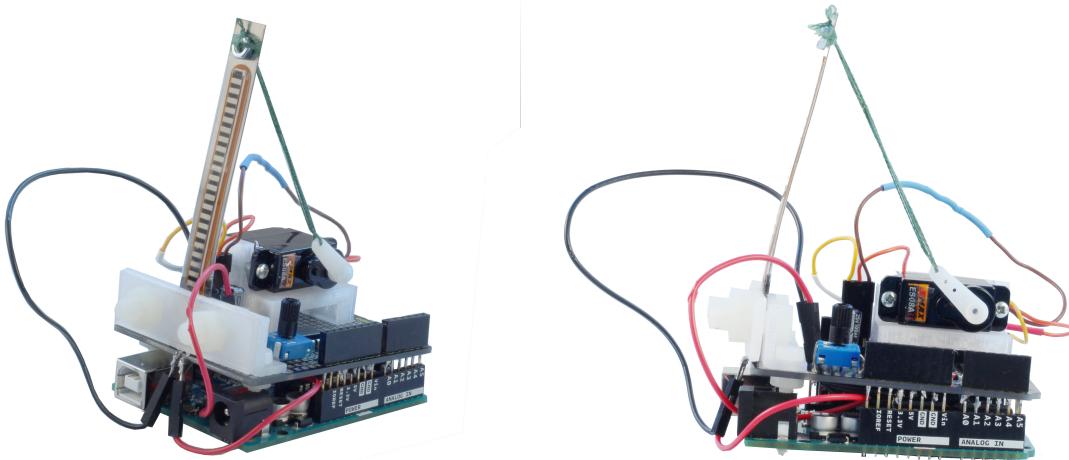
V prvom rade sa kompletne prerobila konštrukcia nosníka s flexi snímačom. Nosník už nemal tvar L a nelebil sa na PCB dosku, na jeho uchytenie sa použili komponenty vytlačené 3D tlačiarňou. Boli to dva L výtlačky, medzi ktoré sa vložil flexi senzor s kovovou časťou a zatiahli sa k sebe skrutkami. Snímač zvierali len v spodnej časti, takže sa mohol v skoro celom svojom rozsahu ohýbať.

Ďalšou veľkou zmenou bola poloha serva. Z predošlého prototypu sa zistilo, že zdvihnutie servomotoru od základnej dosky má nepriaznivý účinok na ohyb snímača a preto sa servo upevnilo čo najbližšie k prototypizačnej doske. Samotný motorček nemá možnosť uchytenia zospodu, len z boku a preto sa musel vytlačiť aj komponent na uchytanie zvlášť tejto súčiastky. Tvar bol navrhnutý tak, aby sa rameno serva mohlo hýbať v rozsahu od 0° do 180° , ale nie viac, pretože pri ohybe nad 180° by sa mohol poškodiť nosík, na ktorom je uchytenu flexi senzor.

Stojan bol v tvaru kvádra s výstupkami a na PCB dosku sa upevňoval zospodu jednou skrutkou. Toto upevnenie nebolo zrovna najvhodnejšie, pretože pri väčšom ohybe nosníka sa skrutka povolila a servo sa pootočilo. Potenciometre, ktoré boli na prvom prototype len dočasným riešením sa nahradili už riadnymi rezistormi s hodnotami $R_1=2,7\text{ k}\Omega$ a $R_2=16\text{ k}\Omega$ [38].

2.1.3 Tretí prototyp

Tretí prototyp sa najviac podobal s finálnou verziou. Hlavnou úlohou v tomto kroku návrhu bolo minimalizovať výtláčky z 3D tlačiarne. Stojan na servo muselo prejsť zmenou tak, aby sa čo najmenšia plocha dotýkala so základnou doskou, plus bola potrebná fixácia nie len jednou, ale dvomi skrutkami. Vytvoril sa mostík, ktorý spravil dostatočný priestor pod sebou pre ostatné komponenty a zafixoval sa v každej nohe jednou skrutkou. Komponenty, čo uchycovali kovový nosník s nalepeným flexi snímačom sa z dvoch L tvarov prekonštruovali na jedno L s prítlačnou doštičkou, čím sa získal o $0,8\text{ cm}$ väčšia vzdialenosť medzi servom a snímačom ohybu. Keďže váha celého shieldu bola veľmi veľká, vymenili



Obr. 2.2: Tretí prototyp [38].

sa kovové skrutky za plastové a po týchto úpravách vážil celý shield len 44 g. Na tretí návrh sa pripájkovali aj kolíky na priame zapojenie do Arduina UNA a potenciometer, ktorý môže študent kreatívne zapojiť do svojho projektu.

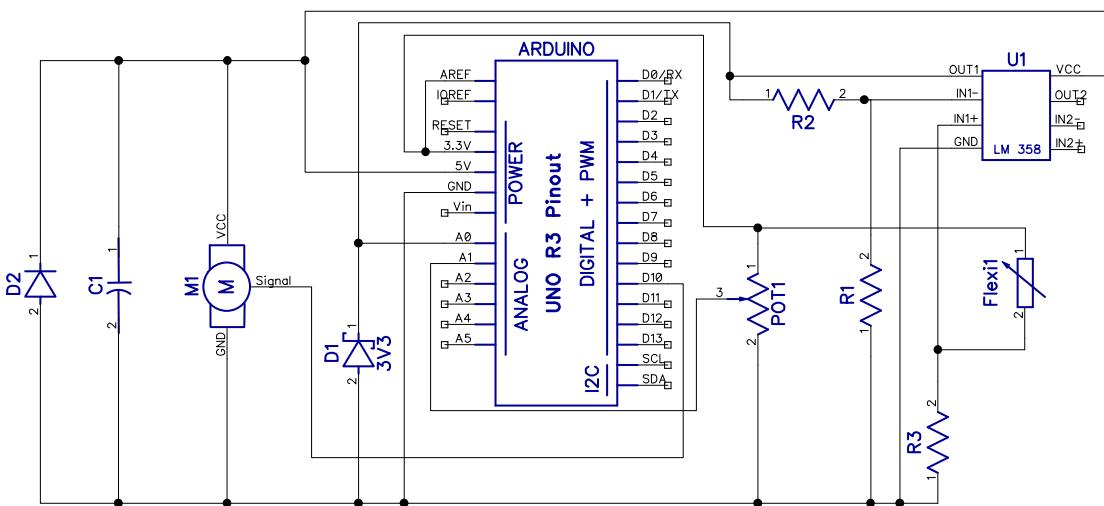
2.1.4 Finálny návrh TugShieldu

Finálny TugShield_R1 výrazne posunul návrh z amatérskej úrovne na poloprofesionálnu. Na výslednom shielde sa komponenty medzi sebou nespájali káblami a ručným pájkovaním, ale navrhla sa plošná doska, ktorá sa poslala do výroby. Zredukovali sa všetky káble, ktoré sa skryli do dosky tenkej 2 mm vyrobenej zo sklenneneho vlákna. Výsledok bol stabilnejší produkt s väčšou odolnosťou voči poškodeniu a v konečnom dôsledku aj s lepším estetickým efektom.

Návrh dosky bol spravený v CAD programe konkrétnie DipTrace [11]. Program obsahuje štyri aplikácie, ktoré zahŕňajú komplexné nástroje na vytváranie plošných dosiek. Hlavný princíp jeho fungovania je vytvorenie elektrickej schémy s komponentom a následne z nej vygenerovať fyzický návrh plošnej dosky.

Avšak nie všetky komponenty z TugShieldu_R1 boli zahrnuté v knižnici komponentov, ktorá je súčasťou DipTrace. Flexi senzor, servomotor a operačný zosilňovač sú nie bežne používané súčiastky a preto prvý krok pri návrhu plošnej dosky bolo zhotoviť najprv tieto diely. Nové súčiastky sa vytvorili v aplikácii Component Editor. Tam sa nakreslila ich elektrická schematická značka a pripájacie piny. Pre servomotor sa zvolila všeobecná schematická značka motorov, ale pre flexi senzory nie je priradená používaná značka, preto ju bolo potrebné navrhnúť.

Podstatou fungovania snímač ohybu je, že mení svoj odpor pri ohybe – deformácií. Súčiastka nie je polarizovaná a do obvodu sa zapája ako odporová napäťová delička. Princíp tohto zapojenia je, že sa za flexi snímačom pripojí ešte jeden rezistor s hodnotou odporu rovnakou, akú má snímač v pokojovom stave a výsledný signál vychádza z bodu medzi snímačom a spomínaným rezistorom. A práve táto podstata fungovania sa musela zohľadniť pri návrhu schematickej značky, ktorá sa nakoniec zvolila ako značka napäťovej deličky, rovnaká sa používa aj pri potenciometri.



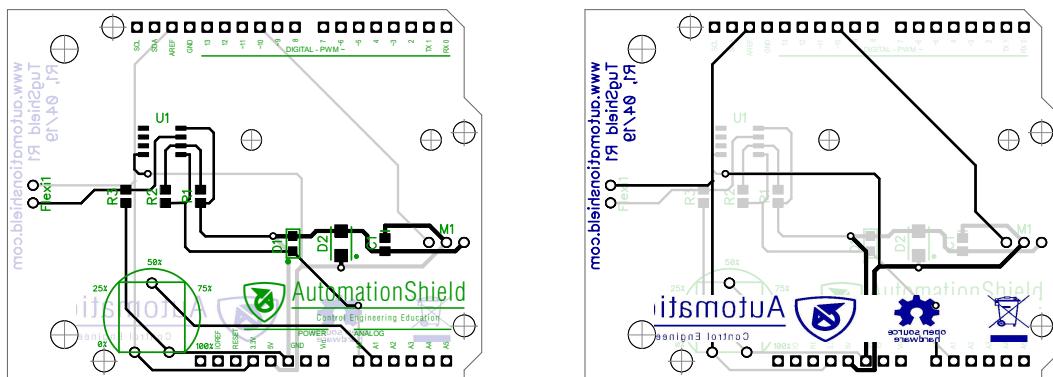
Obr. 2.3: Schéma zapojenia [38].

Posledná súčiastka, ktorá sa nenachádzala v knižnici, bol operačný zosilňovač. Značka sa navrhla ako jednoduchý obdlížnik s pinmi na pripojenie. Ku všetkým schematickým

značkám sa muselo vytvoriť aj ich fyzické rozhranie v Pattern editore. Podľa dátových hárkov od výrobcov sa presne navrhli plôšky na napájkovanie a spojili sa so súborom schematickej značky súčiastky.

Mohlo sa prejsť k návrhu celkovej dosky. Nosným komponentom bolo Arduino Uno a ostatné súčiastky sa kategorizovali do dvoch skupín. Prvá skupina bola tvorená hlavným komponentom a to servomotorom, ku ktorému bola zapojená dióda pre ochranu Arduina pred spätnými prúdmi a kondenzátor s kapacitou $100 \mu\text{F}$ na vyhladenie signálu zo serva [38].

Druhú skupinu tvoril flexi senzor a operačný zosilňovač. Flexi snímač bol pripojený na 5 V a výstupný signál z neho nevyužíval celú škálu ADC úrovní. Preto bolo nutné ho zosilniť pomocou operačného zosilňovača, takzvaného opampa. Opamp bol zapojený do neinvertujúceho zapojenia [1], teda signál zo senzoru sa pripojil na pin IN+, pin IN- sa uzemnil cez rezistor R_1 a výstup (OUT pin) s IN- spojil rezistor R_2 . Pomerom rezistorov sa dala nastavovať veľkosť zosilnenia signálu a takýto zosilnený signál sa mohol pripojiť na Arduino analógový pin A0. Súčiastka LM358ADR obsahuje nie len jeden, ale dva operačné zosilňovače, pričom v TugShieldu_R1 sa druhý opamp nevyužil.



Obr. 2.4: Predná a zadná strana finálnej PCB dosky [38].

Skôr, ako sa signál pripojil na pin A0, musel prejsť cez zenerovu diódu. Tá mala za úlohu obmedziť veľkosť napäťia vstupujúceho do Arduina a konkrétny cieľ bol, aby napätie nepresiahlo hodnotu 3,3 V a tak bolo kompatibilné aj s inými zariadeniami ako napríklad Arduino Zero a Metro. Tieto zariadenia majú rovnaké rozhranie pinov triedy R3, ale ich MCU pracujú na logike 3,3 V a nie 5 V. Pôvodný plán bol napájať rovno flexi snímač aj opamp na 3,3 V, ale operačný zosilňovač LM358ADR je od výrobcu vyhotovený na 5 V.

Program automaticky konvertoval schému na PCB dosku a v aplikácii PCB Layout [11] sa komponenty rozmiestnili na dosku tvaru Arduino Uno tak, aby nebránili nainštalovaniu 3D výtlačkov. Takto hotová doska sa poslala do výroby a po vyhotovení sa na ňu jednoducho napájkovali súčiastky, pričom celková cena dosky aj s komponentami bola len 15,68€[38].



Obr. 2.5: Finálny prototyp [38].

2.2 Softvér TugShield_R1

Druhou, rovnako dôležitou časťou záverečnej práce bolo vytvorenie API (programátor-ského rozhrania aplikácie) pre TugShield_R1. Síce sa shield dal po pripojení do počítača plne ovládať cez aplikáciu Arduino IDE, bolo potrebné navrhnúť aj viac prístupnú formu pre užívateľov.

2.2.1 TugShield trieda

V jazyku C/C++ sa vytvorila trieda s názvom **TugShield**, ktorá bola zaradená v súbore **TugShield.h**. Trieda **TugShield** obsahovala štyri metódy na ovládanie hardvéru TugShiledu_R1, plus dodatočné funkcie ako zabránenie opäťovnému volaniu knižnice, zahrnutie iných potrebných knižníc či definovanie konštant, ktoré sú potrebné na fungovanie [38].

```
class TugShieldClass
{
public:
    void begin();
    void calibration();
    void actuatorWrite(int servo_angle);
    float sensorRead();
```

Metóda begin

Ako je už z názvu jasné, tak metóda **begin** mala za úlohu zinicializovať hardvér po pripojení do počítača. V metóde sa nachádzali len tri príkazy a to zadefinovanie pinu servo motora **servo.attach**, nastavenie ramena serva na pozíciu nula **servo.write** a externej analógovej referencie **analogReference**.

```
void TugShieldClass::begin() {
    servo.attach(TUG_UPIN);
    servo.write(SERVO_MAX);
```

```
analogReference(EXTERNAL);}
```

Metóda calibration

Metóda `calibration` bola druhým krokom po pripojení TugShieldu_R1 k PC [38]. Shield sa kalibroval v dvoch koncových bodoch, kde sa získavala maximálna a minimálna hodnota signálu z flexi snímača. Najprv sa rameno serva nastavilo na polohu 0° , kde snímač neboli vobec ohnuty. Meranie signálu sa vykonalo $10 \times$, z čoho sa vybrala najmenšia nameraná hodnota. Ten istý postup sa opakoval aj pri maximálnom ohnutí ramena serva.

```
void TugShieldClass::calibration() {
    Serial.println("Prebieha kalibracia");
    servo.write(SERVO_MAX);
    delay(DELAY_VALUE);
    for (int ii=0; ii <= 10; ii++) {
        _sensorRead=analogRead(TUG_YPIN);
        if (ii == 00) {
            if(_sensorRead>K_MAX) {
                k_maximal=_sensorRead;
            }
            delay(DELAY_VALUE);
        }
        else {
            if(_sensorRead>k_maximal) {
                k_maximal=_sensorRead;
            }
            delay(DELAY_VALUE);
        }
    }
}
```

Metóda actuatorWrite

`ActuatorWrite` nastavuje uhol ramena servomotora a je jediná metóda, ktorá vyžadovala vstup od používateľa. Užívateľ musel zadat želaný uhol, ktorý sa následne prepočítal do hodnoty, ktorá sa nastavila na servo motor [38].

```
void TugShieldClass::actuatorWrite(int servo_angle) {
    int servo_rightangle = 180 - servo_angle;
    servo.write(servo_rightangle);
}
```

Metóda sensorRead

Metóda `sensorRead` bola najkomplikovanejšia metóda, ktorá má na starosti prečítať a zároveň premapovať signál zo snímaču ohybu [38]. V prvom rade sa teda prečíta aktuálna hodnota z flexi snímača pomocou príkazu `analogRead` a potom sa skontroluje podmienka či bola vykonaná kalibrácia metódou `calibration`. Ak bola kalibrácia uskutočnená v predošlých krokoch, dátá sa premapujú podľa minima a maxima z kalibrácie.

Ak neboli TugShieldu_R1 nakalibrovaný, na premapovanie sa použijú vopred nastavené hodnoty.

```
float TugShieldClass::sensorRead() {
    _sensorRead = analogRead(TUG_YPIN);
    float _sensorValue;
    if(_wasCalibrated == true) {
        _sensorValue = AutomationShield.mapFloat
            ((float)_sensorRead,(float)k_minimal,
             (float)k_maximal, 0.00, 100.00);
    }
    float _sensor_rightValue = 100 - _sensorValue;
    return _sensor_rightValue;}
```

2.2.2 Demonštračné príklady

Na overenie fungovania hardvéru a triedy `TugShield` sa vytvorili demonštračné príklady v aplikácií Arduino IDE. Výstupom z príkladu sú vstupno-výstupné dátá, ktoré sa ďalej môžu využiť.

SelfTest

Príklad `SelfTest` bol najjednoduchší príklad zo série príkladov. Jeho cieľom bolo skontrolovať kalibráciu TugShieldu_R1.

Na začiatku programu sa vykoná inicializácia metódou `TugShield.begin` a následne na to kalibrácia pomocou `TugShield.calibration`. Na obrazovke sa vypíše oznam o spustení merania a rameno servomotora sa nastaví na polohu 0° cez `TugShield.actuatorWrite`. Hodnoty merania sa zistia metódou `TugShield.sensorRead` a program vyhodnotí či sa nachádza v danom intervale. Interval v podmienke bol zdefinovaný od $0\text{--}15\%$ [38].

```
Serial.println("Testovanie flexi snimaca... poloha serva: 0");
TugShield.actuatorWrite(0);
delay(500);
y = TugShield.sensorRead();
if (y >= 0.0 && y <= 15.0) {
    Serial.println("V poriadku.");
    Serial.println(y);
} else {
    Serial.println("Nevyhovuje.");
    Serial.println(y); }
```

Rovnaký postup sa opakoval aj pri ohnutí ramena servomotora do maximálnej polohy 180° . Pre splnení podmienky sa hodnoty museli nachádzať v intervale od $85\text{--}100\%$ [38].

Výstupom tohto príkladu bol výpis na monitor sériového portu, ktorý obsahoval informáciu o začatí a skončení testu, výsledky pri obidvoch oblastiach testovania a aj s príslušnými nameranými hodnotami.

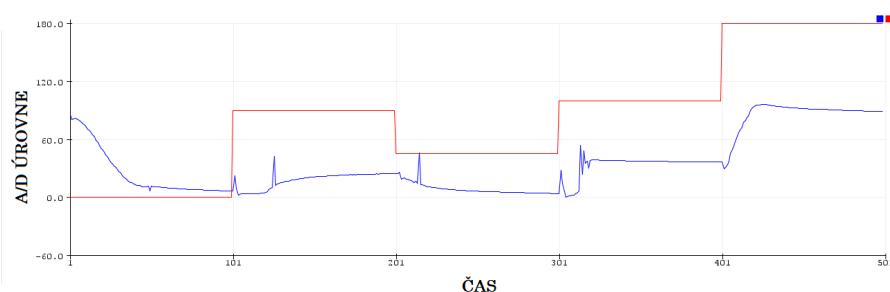
Identifikácia systému

Ďalším príkladom, na ktorom sa dali získať vstupno-výstupné dátá, bola Identifikácia systému. Princíp bol vo vytvorení súboru preddefinovaných vstupov a celý príklad bežal v otvorenej slučke.

Rovnako ako v `SelfTest` sa musel systém na začiatku inicializovať `TugShield.begin()` a kalibrovať `TugShield.calibrate()`. Definovali sa premenné ako T_s – períoda vzorkovania, k – počet vzoriek, `enable` – inicializácia potvrdenia kroku, ktoré sa využili v podsystéme vzorkovania z knižnice `AutomationShield` [38]. V hlavnej slučke programu sa opakoval vždy postup:

- 1. Ak prešiel dostatočný čas na ďalšiu vzorku vykonaj krok.
- 2. V kroku sa vykonalo meranie.
- 3. Počkalo sa na ďalšiu vzorku.

```
void setup() {
    Serial.begin(9600);
    TugShield.begin();
    TugShield.calibration();
    Sampling.period(Ts * 1000);
    Sampling.interrupt(stepEnable);}
void loop() {
    if (enable) {
        step();
        enable=false; }
void stepEnable(){
    enable=true; }
void step(){
    if (k % (T*i) == 0){
        u = U[i];
        i++; }
    TugShield.actuatorWrite(u);
    y = TugShield.sensorRead();
    k++; }
```



Obr. 2.6: Výstup z príkladu identifikácie systému [38].

Postup sa realizoval cez funkcie `void step()` a `void stepEnable()`, kde funkcia `stepEnable()` kontroluje či prešiel dostatočný čas a pomocou bool premennej enable spúšťa funkciu `step()`, zapísanie uhla do serva a odčítanie hodnoty zo snímača [38].

Aplikovanie PID regulátora

Posledný charakteristický príklad v záverečnej práci mal za úlohu riadiť spätnú väzbu ohybu nosníka s využitím PID regulátora. Štruktúra programu bola veľmi podobná ako v príklade identifikácia systému a tiež tu figurovala vopred určená trajektória vstupov.

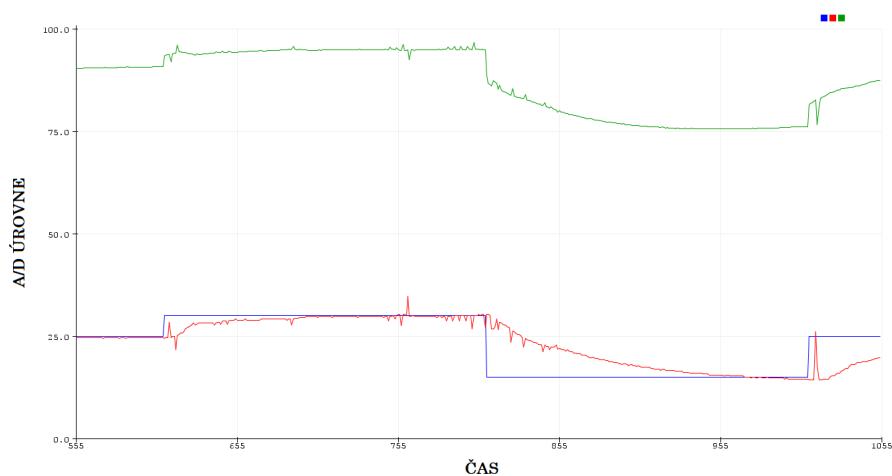
Okrem premenných, ktoré sa vyskytovali v príklade pred tým, sa museli pridať ešte štyri nosné premenné a to: TI, TD, KP a r. Premenná r vyjadruje žiadanú hodnotu deformácie a TI, TD a KP sú regulačné konštanty PID regulátora. Tieto konštanty sa určili len experimentálne, metódou pokusov a opráv. Výsledky príkladu jasne dokázali, že systém je lineárny v rozpätí ramena servomotora od 0° – 35° a od 35° a viac sa systém nedal regulovať jednoduchým PID regulátorom, pretože bol nelineárny [38].

```

void setup() {
    PIDAbs.setKp(KP);
    PIDAbs.setTi(TI);
    PIDAbs.setTd(TD); }

void loop() {
    if (enable) {
        step();
        enable=false;}}
void stepEnable(){
    enable=true; }
void step(){
    if (k % (T*i) == 0){
        r = R[i]; i++;}

u = PIDAbs.compute(r-y,0,180,0,100);
    
```



Obr. 2.7: Výstup z príkladu PID [38].

3 Vývoj nového hardvéru

Prvá verzia TugShieldu prešla dlhým vývojom hlavne v oblasti hardvéru, ale aj napriek tomu sa na výslednej verzii objavil priestor na zlepšenie.

Flexi snímač na TugShield_R1 bol zapojený do deličky napäťia v poradí najprv flexi snímač a až potom rezistor. Toto poradie spôsobovalo, že pri pozícií ramena servomotora 0° signál z flexi snímača dosahoval maximum a pri ohýbaní klesal. Pre ďalšie použitie to nebolo ideálne a preto sa signál prevrácal softvérovo, aby pri ohýbaní nosníka stúpal. Tento problém jednoducho vyriešila zmena poradia v deličke. Najprv sa umiestnil rezistor a potom sa pripojil snímač ohybu.

Aby bol TugShiled_R1 kompatibilný nie len s Arduinom Unom pracujúcim na 5 V logike, ale aj s ostnými mikroradičmi s hardvérovou verziou rozdelenia pinov R3, ktoré po väčšine majú len maximálne napätie 3,3 V, muselo sa zabezpečiť, aby spätné napätie do mikroradiča nebolo väčšie ako 3,3 V. Operačný zosilňovač, ktorý bol použitý na TugShield_R1 LM358 mal napájanie 5 V a nebol rail-to-rail. Na výstupe teda bolo napätie menšie ako 5 V, ale nebolo zaručené, že bolo aj menšie ako 3,3 V. O tento problém sa postarala zenerova dióda, ktorá sa umiestnila na konečný výstup a neprepustila napätie väčšie ako 3,3 V.

Na TugShielde_R2 sa celý problém zobrajal z iného uhlhu pohľadu a opamp LM258 sa nahradil TLC2272, ktorý je už priamo napájaný z 3,3 V. Zenerova dióda tak nebola potrebná v novom obvode. Navyše opamp TLC2272 má aj benefit, že je rail-to-rail, čo je vlastnosť operačných zosilňovačov, ktorých dynamický rozsah je schopný dosiahnuť extrémy napájacieho napäťia [17].

Ďalším nedostatkom bolo zapojenie operačného zosilňovača. Na TugShield_R1 sa použilo neinvertujúce zapojenie opampu, ktoré vykonáva funkciu impedančného meniča s veľkým vstupným a malým výstupným odporom [1] a výsledné zosilnenie vyplýva z Rov. (3.1).

$$A_u = U_{out}/U_{in} \quad (3.1)$$

$$U_{in} = U_{R_1}$$

$$U_{in} = I \cdot R_1$$

$$U_{out} = U_{in} + I \cdot R_2$$

$$U_{out} = I \cdot R_1 + I \cdot R_2$$

$$A_u = (I \cdot R_1 + I \cdot R_2)/I \cdot R_1$$

$$A_u = 1 + R_2/R_1 \quad (3.2)$$

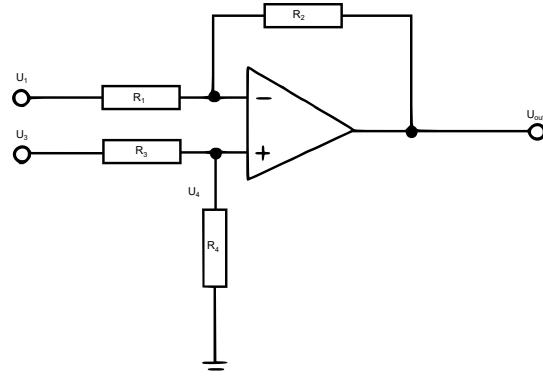
kde

- U_{out} – výstupné napätie [V],

- A_u – konečné zosilnenie [V/V],
- U_{in} – vstupné napätie [V].

Toto zapojenie ale nebola najvhodnejšia voľba pretože, sa ním nedokázalo docieliť, aby signál využíval všetky dostupné úrovne ADC aj keď sa použili akékoľvek hodnoty rezistorov.

Preto sa pre nový model – TugShield_R2 zvolilo diferenčné zapojenie operačného zosilňovača. Tento typ zapojenia funguje na trochu zložitejšom princípe a to že na obidva vstupy sa musí priviesť vstupné napätie a výstupné napätie je dané rozdielom obidvoch vstupov. Rozdiel vstupov sa zosilní pomerom rezistorov, ktoré vyplýva z Rov.(3.3). Základnú schému zapojenia je možné vidieť na Obr. 3.1.



Obr. 3.1: Diferenčné zapojenie operačného zosilňovača.

$$(U_1 - U_4)/R_1 = (U_4 - U_{out})/R_2 \quad (3.3)$$

$$U_4/R_4 = (U_3 - U_4)/R_3 \quad (3.4)$$

$$R_2 \cdot U_1 - R_2 \cdot U_4 = R_1 \cdot U_4 - R_1 \cdot U_{out}$$

$$R_3 \cdot U_4 = R_4 \cdot U_3 - R_4 \cdot U_4$$

$$(U_1 \cdot R_2 + U_{out} \cdot R_1)/(R_1 + R_2) = U_3 \cdot R_4/(R_3 + R_4)$$

$$U_3 \cdot R_4 \cdot (R_1 + R_2) = U_1 \cdot R_2 \cdot (R_3 + R_4) + U_{out} \cdot R_1 \cdot (R_3 + R_4)$$

$$U_{out} \cdot R_1 \cdot (R_3 + R_4) = U_3 \cdot R_4 \cdot (R_1 + R_2) - U_1 \cdot R_2 \cdot (R_3 + R_4) \quad (3.5)$$

$$U_{out} = (U_3 \cdot R_4 \cdot (R_1 + R_2) - U_1 \cdot R_2 \cdot (R_3 + R_4))/(R_1 \cdot (R_3 + R_4)) \quad (3.6)$$

Ak položíme $R_1 = R_3$ a $R_2 = R_4$ potom:

$$U_{out} = R_2/R_1(U_3 - U_1) \quad (3.7)$$

kde

- U_{out} – výstupné napätie [V],
- U_3 – vstupné napätie 2 [V],
- U_1 – vstupné napätie 1 [V],

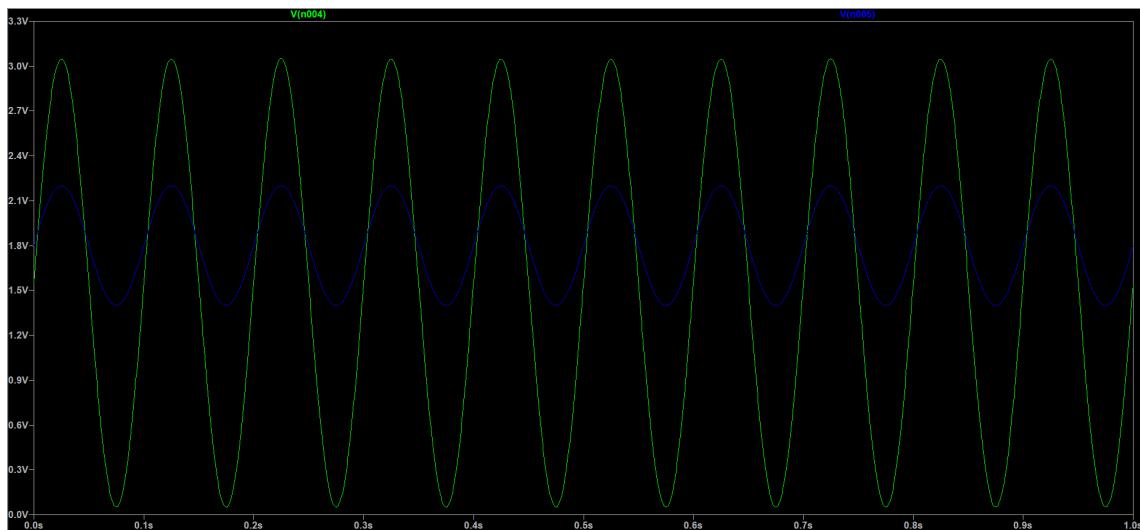
- R_1 – odpor na rezistore č.1 [Ω],
- R_2 – odpor na rezistore č.2 [Ω],
- R_3 – odpor na rezistore č.3 [Ω],
- R_4 – odpor na rezistore č.4 [Ω].

V prípade TugShieldu_R2 predstavuje U_3 vstupné napätie z flexi snímača, U_1 napätie, ktoré bude znižovať offset od nuly a U_{out} výsledné zosilnené napätie.

3.1 Modelovanie v LTSpice

Pri vývoji návrhoch zapojenia sa využívala simulácia na overenie funkčnosti obvodu a na naladenie operačného zosilňovača. Zapojenia sa simulovali v programe LTSpice.

LTspice je vysoko výkonný simulačný softvér SPICE, ktorý pracuje s elektrickými schémami a modelmi, zobrazuje krivky a simuláciu analógových obvodov. Súčasťou LTspice sú makromodely pre väčšinu spínacích regulátorov, zosilňovačov a knižnica zariadení na simuláciu všeobecných obvodov. Vďaka vylepšeniam tohto softvéru je simulácia spínacích regulátorov extrémne rýchla v porovnaní s normálnymi SPICE simulátormi, čo umožňuje používateľovi zobraziť krivky väčšiny regulátorov za pár minút [2].



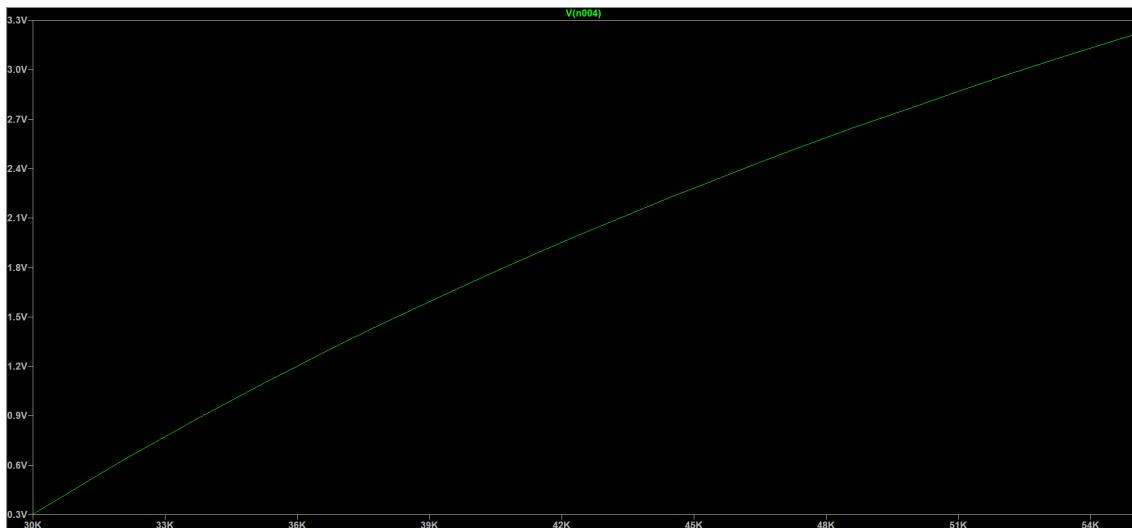
Obr. 3.2: Diferenčné zapojenie s ideálnymi zdrojmi.

Prostredie vo free verzií poskytuje knižnicu základných komponentov a nástrojov, ktoré sa používajú na vytvorenie obvodov. Na základnom paneli sa nachádzajú najpoužívanejšie funkcie, ktoré sa aktivujú jednoducho “drag and drop”.

V obvode pre TugShield_R2 sa nachádza len jeden komponent, ktorý nie je v knižnici LTSpice - operačný zosilňovač TLC2272. Model tejto súčiastky sa musel najprv importovať z externého zdroja. Súbor s modelom má príponu .LIB a cesta k nemu sa zadáva cez programovacie okno pod ikonkou .op. V knižnici komponentov sa vyhľadal univerzálny operačný zosilňovač a do jeho hodnoty sa zapísal názov súboru externého modelu. Takýto

upravený opamp sa použil ako základ obvodu. K nemu sa pripojili 4 rezistory do diferenčného zapojenia. Ako vstupné napäťia do opampu sa použili v prvej simulácii ideálne zdroje, ktoré ale nezachytávali úplne skutočné vlastnosti reálnych signálov.

V programe LTSpice sú rôzne druhy simulácií. Prechodová analýza zobrazí parameter ako prúd alebo napätie v čase. Na výstupe je možné vidieť správanie počas zadanej doby. AC analýza poskytuje frekvenčnú odozvu obvodu. Výstupný priebeh je bodový diagram, ktorý ukazuje amplitúdu a fázu v určenom frekvenčnom rozsahu. Existuje niekoľko možností s AC analýzou. Frekvenčná odozva sa môže zobraziť ako bodový graf na karteziánskej súradnicovej rovine so skutočnou a imaginárnu osou alebo ako Nyquistov graf. DC Sweep je typ simulácie, ktorá umožňuje meniť napätie alebo prúd konkrétneho zariadenia. Na určitých komponentoch z knižnice je zadaný rozsah napäťia, pre ktorý môže produkt bezpečne fungovať. Analýza šumu pri správnom modelovaní umožňuje zobraziť vlastný šum systému a tiež šum z vonkajšieho zdroja a funkcia DC Transfer počíta nízkofrekvenčný zisk a vstupný a výstupný odpor obvodu [35].



Obr. 3.3: Diferenčné zapojenie s reálnym signálom z flexi snímača.

V prvej simulácii s ideálnymi zdrojmi sa použila prechodová analýza zobrazujúca zmenu napäťia v čase. Výstup zo simulácie boli dve sínusoidy - pôvodné nezosilnené napätie flexi snímača a výstupné zosilnené napätie. Hodnoty rezistorov sa zvolili tak, aby pôvodný signál bol zosilnený a išiel od 0–3,3 V čo najpresnejšie. Simulácia je zobrazená na Obr. 3.2.

V druhej simulácii bol zdroj, ktorý simuloval signál z flexi snímača nahradený deličkou napäťia. Flexi snímač sa nasimuloval ako rezistor s premenlivým odporom. Tento raz sa ale simulácia nezobrazila v čase, ale ako závislosť na rastúcom odpore v premenlivom rezistore. Avšak simulácia nefungovala správne a po dlhom uvážení sa prišlo na príčinu. Signál z deličky napäťia bol napojený cez rezistor R_2 do opampu, ale to vytvára ešte jednu nechcenú deličku a v obvode vznikajú impedancie. Preto sa musel využiť ešte jeden operačný zosilňovač, cez ktorý prešiel signál z flexi snímača bez zosilnenia a až potom sa napojil na rezistor do opampu 2.

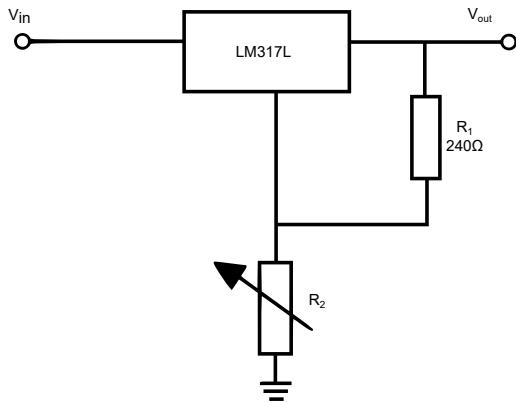
Na Obr. 3.3 je zobrazený zosilnený signál, ktorý ide približne od 0,2 V do 3,2 V čo je ideálne zosilnenie.

3.2 Riešenie problému vstupného napäťia

Stále ale ostávala otvorená otázka akým spôsobom sa priviedie na diferenčný operačný zosilňovač druhé vstupné napätie na odstránenie offsetu od nuly. Predošlou simuláciou sa zistilo, že hodnota napäťia by mala byť približne od 1,3–1,6 V, ktorá sa ale nebude meniť v čase.

Prvá idea bola použiť jednoduchú deličku napäťia. Do obvodu sa zapojili dva rezistory s hodnorou $R_1=1\ \Omega$ a $R_2=1,3\ \Omega$, ktoré by vytvárali požadovanú hodnotu. Tento nápad sa najprv testoval samostatne, kde fungoval, ale potom sa otestoval aj v reálnom obvode a vznikal rovnaký problém ako s deličkou napäťie s flexi snímačom.

Preto druhý nápad bolo na dosiahnutie napäťia – napäťový regulátor. Podľa parametrov sa vybrala súčiastka LM317, ktorej výstupné napätie sa dalo zregulovať v rozmedzí od 1,2 V do 37 V. Regulátor má tri piny V_{in} , V_{out} a Adjust. Na V_{in} sa priviedlo napájanie z Arduina 3,3 V, Adjust sa spojil cez rezistor s odporom $240\ \Omega$ so V_{out} a zároveň sa cez rezistor R_2 uzemnil.



Obr. 3.4: Aplikácia LM317.

Pri zmene veľkosti odporu rezistora R_2 sa nastavuje celková regulácia napäťia podľa Rov. (3.8), pričom ak I_{Adj} je menšie ako $100\ \mu\text{A}$, druhá zložka vzťahu je zanedbateľná, čo platí aj v prípade TugShield_R2. Výsledná regulácia sa teda vypočíta zjednodušene pomocou Rov. (3.9).

$$V_{out} = 1,25V(1 + R_2/R_1) + I_{adj}R_2 \quad (3.8)$$

$$V_{out} = 1,25V(1 + R_2/R_1) \quad (3.9)$$

Zapojenie a naladenie regulátora sa nesimulovalo, ale samostatne sa otestovalo na breadboarde. Vyskúšali sa rôzne kombinácie rezistorov, ktoré sa mohli použiť potom do celkového obvodu Tugshield_R2. Odpor R_1 zostal na hodnote, ktorá bola odporúčaná od výrobcu $240\ \Omega$, menil sa len odpor rezistora R_2 . Merala sa výsledné napätie na pine V_{out} . Pri odpore rezistora R_2 $47\ \Omega$ bol výsledok $1,55\ \text{V}$ a pri $R_2=68\ \Omega$ $1,6\ \text{V}$. Overilo sa to aj dosadením do Rov.(3.9), kde pri $R_2=47\ \Omega$ sa $V_{out}=1,4948\ \text{V}$ a pri $R_2=68\ \Omega$ $V_{out}=1,604$

V. Rozdiely sa môžu považovať za zanedbateľné a mohli byť spôsobené odporom káblu, s ktorými vzťah nepočíta.

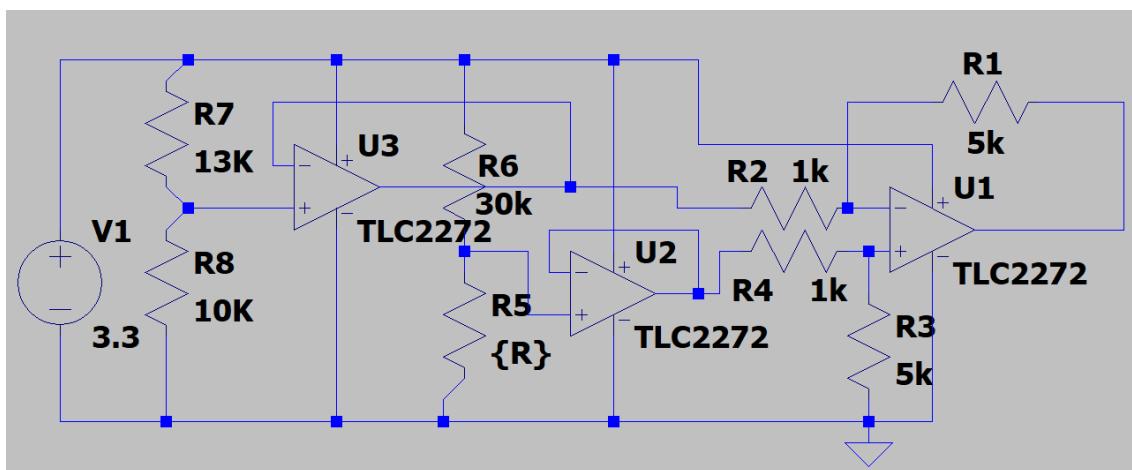
Celá časť sa napájkovala aj so zvyšným obvodom na prototypizačnú dosku arduina a doska sa otestovala. Po pripojení do počítača sa vyskúšal jednoduchý program na test signálov a či je správne naladené zosilnenie. Zistilo sa, že signál zosilnený opampom a privedený na pin A1 sa nemenil, nech bola pozícia ramena servomotoru akákoľvek.

Pre zistenie príčiny tohto problému sa museli vyskúšať jednotlivé časti obvodu. Keďže napäťový regulátor s rezistormi bol už odokončaný samostatne predtým, vyskúšala sa ešte samostatne aj časť obvodu tvorená operačným zosilňovačom a flexi senzorom. Aj tá fungovala a teda problém bol pri spojení týchto dvoch častí dokopy. Pre overenie teórie sa na miesto napäťového regulátora použil osciloskop Analog Discovery 2.

Analog Discovery 2 je USB osciloskop, logický analizátor a multifunkčný prístroj, ktorý umožňuje používateľom merať, vizualizovať, generovať, zaznamenávať a riadiť obvody zmiešaného signálu všetkých druhov. Je mimoriadne kompaktný, ale výkonný a plnohodnotne nahradza množstvo labortórneho vybavenia. Analógové a digitálne vstupy a výstupy je možné pripojiť k obvodu pomocou drôtových sond alebo použiť iné adaptéry. S využitím bezplatného softvéru WaveForms, ktorý je kompatibilný s Mac, Linux aj Windows, môže byť Analog Discovery 2 nakonfigurovaný tak, aby fungoval ako ktorýkoľvek z niekoľkých tradičných testovacích a meracích prístrojov vrátane osciloskopu, generátora kriviek, napájacieho zdroja, voltmetra, záznamníka dát, logického analizátora a analizátoru impedancie [10].

Pomocou tohto, zariadenia sa vygenerovalo napätie 1,5 V, ktoré sa napojilo na pin IN- diferenčného operačného zosilňovača. Spustil sa rovnaký testovací program ako v experimente pred tým, s tým rozdielom, že teraz signál reagoval na ohyb nosníka a bol ideálne zosilnený.

Po úvahách sa prišlo na to, že problém, ktorý nastal, keď napätie bolo vytvorené deličou napäťia dvoch rezistorov, nevyriešil ani napäťový regulátor LM317. Princíp problému zostáva taký istý, keďže LM317 rovnako využíva rezistory na uregulovanie napäťia z Arduina. Vytvára sa nechcená delička, ako to bolo aj pri flexi snímači a rovnako ako aj v tom prípade, sa tento problém môže odstrániť ďalším operačným zosilňovačom podľa nasledujúcej schémy Obr. 3.5.



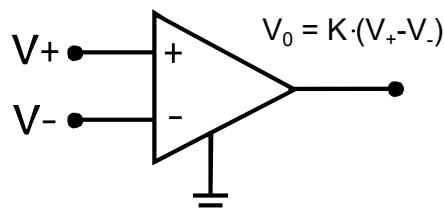
Obr. 3.5: Schéma zapojenia s 3 operačnými zosilňovačmi.

Návrh zapojenia je teda nasledovný: V centre obvodu sa nachádza diferenčný operačný zosilňovač TLC2272, ktorý je napájaný 3,3 V a je rail-to-rail. Tak isto z 3,3 V je napájaná delička napäťa obsahujúca flexi snímač (v schéme premenlivý rezistor R) a rezistor $30\text{ k}\Omega$, odkiaľ putuje signál do prvého buffer opampu a až potom do centrálneho operačného zosilňovača. Na druhý pin IN- je privádzané napätie 1,5 V pomocou deličky napäťa s rezistormi $R_1=1,3\text{ k}\Omega$ a $R_2=1\text{ k}\Omega$. Toto napätie však ešte pred pripojením na IN- prejde cez buffer opamp, aby sa dosiahlo správny výsledok fungovania celého obvodu TugShieldu.

Tento návrh by rovnako fungoval, aj keby sa napätie regulovalo pomocou napäťového regulátora LM317 aj cez deličku napäťa. Kvôli šetreniu miesta na TugShielde sa vybrala možnosť s obyčajnou deličkou napäťa, pretože sa na dosku bude musieť umiesť nie len jedna súčiastka TLC2272, ale dve, pretože každá z nich obsahuje 2 kusy operačných zosilňovačov a na tento návrh sa použijú až tri.

3.3 Návrh obvodu podľa základných princípov operačných zosilňovačov

Na celú problematiku sa ale pozrelo aj zo začiatocného bodu a to či diferenčné zapojenie opampu bola dobrá voľba a či by sa celý obvod nedal zjednodušiť. Ako inšpiráciu pre nasledujúci návrh sa zobraza prednáška z Massachusetts Institute of Technology[24], kde je vysvetlený základ fungovania operačných zosilňovačov.

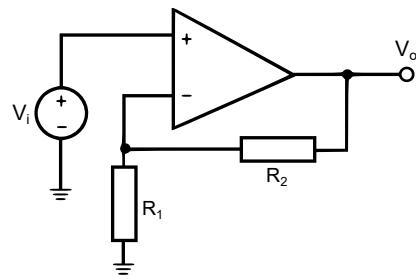


Obr. 3.6: Model ideálneho operačného zosilňovača.

Ideálny operačný zosilňovač pracuje na princípe, že konenčné zoislnenie V_0 je dané rozdielom vstupov a násobené konštantou K . Zjednodušene by sa model opampu dal vyjadriť podľa Obr. 3.6, kde V_+ predstavuje napätie, ktoré sa zosilňuje. Zosilnenie si pri správnom zapojení možno prestaviť ako pridanie nahromadeného elektrického náboja k vstupnému V_+ napätiu, čím vzniká zosilnené V_0 napätie.

Pokiaľ sa do obvodu umiestnia rezistory R_1 a R_2 (Obr. 3.7), zo základného vzťahu pre opampy sa dá jednoducho vyjadriť pomer vstupného napäcia V_i a výstupného V_0 Rov. (3.10).

$$\begin{aligned}
 V_+ &= V_i \\
 V_- &= R_1 \cdot V_0 / (R_1 + R_2) \\
 V_0 &= K \cdot (V_+ - V_-) \\
 V_0 &= K \cdot (V_i - R_1 \cdot V_0 / (R_1 + R_2)) \\
 V_0/V_i &= (R_1 + R_2)/R_1 *
 \end{aligned} \tag{3.10}$$



Obr. 3.7: Ideálny operačný zosilňovač s dvomi rezistormi.

*platí, ak K je veľká hodnota

Pomocou týchto poznatkov sa vytvorila úplne nová schéma, ktorá bola mimoriadne jednoduchá. Flexi snímač sa umiestnil do deličky napäťa, ktorej ale druhý rezistor mal odpor mnohonásobne vyšší ako odpor flexi snímača. Napätie sa odtiaľto presunulo na pin IN+ operačného zosilňovača. V druhej vetve obvodu sa tiež pomcoou deličky napäťa vytvorilo druhé, veľmi malé napätie, ktoré sa ale nemenilo pri ohýbaní. Cez buffer opamp a rezistor R_1 sa pripojilo na pin IN- a na záver sa výstup V_0 spojil s IN- cez rezistor R_2 . Takáto schéma sa nasimulovala (Obr. 3.8).



Obr. 3.8: Simulácia návrhu obvodu s jednoduchou logikou.

3.4 Tvorenie PCB dosky v programme DipTrace

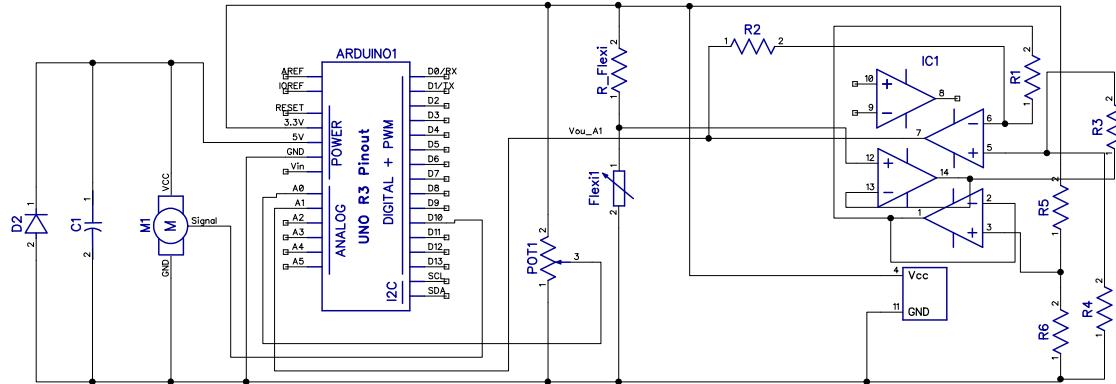
Vznikli teda dva funkčné finálne návrhy TugShield_R2A (s 3-mi operačnými zosilňovačmi) a TugShield_R2B (jednoduchšia schéma s 2-mi operačnými zosilňovačmi). Pre obidva sa rozhodlo, že sa pošlú do výroby a preto nasledujúcim krokom bolo vytvoriť schému v CAD prostredí.

Využil sa program DipTrace, ktorý umožňuje vytváranie schematických diagramov a dosiek plošných spojov. Softvér má štyri hlavné nástroje:

- Schematic Capture - vytváranie elektrických schém,
- PCB Layout - tvorba PBC dosky,
- Component Editor - editor schematických značiek komponentov,
- Pattern Editor - návrh fyzického rozhrania súčiastok.

Súčiastky ako operačné zosilňovače, flexi senzor, servomotor a ďalšie, ktoré neboli zahrnuté vo free verzií knižnice sa nakreslili pomocou Component Editoru a Pattern Editoru.

Arduino UNO, potenciometer a originál grafika projektu tiež neboli obsiahnuté v pôvodnej DipTrace knižnici, ale už boli pripravené v knižnici AutomationShield a len sa použili.



Obr. 3.9: Schéma zapojenia TugShield_R2A.

Ako prvý krok sa musela zestrojiť schéma elektrického zapojenia v Schematic Capture. Pre obidve schémy bol centrálny prvok Arduino UNO a komponenty sa rozmiestňovali na ľavo alebo pravo podľa toho, do akej skupiny patrili, teda s akými ďalšími komponentami boli spojené.

Lavá strana bola v oboch prípadoch rovnaká. Umiestnil sa sem servomotor s usmerňovacou diódou a kondenzátorom.

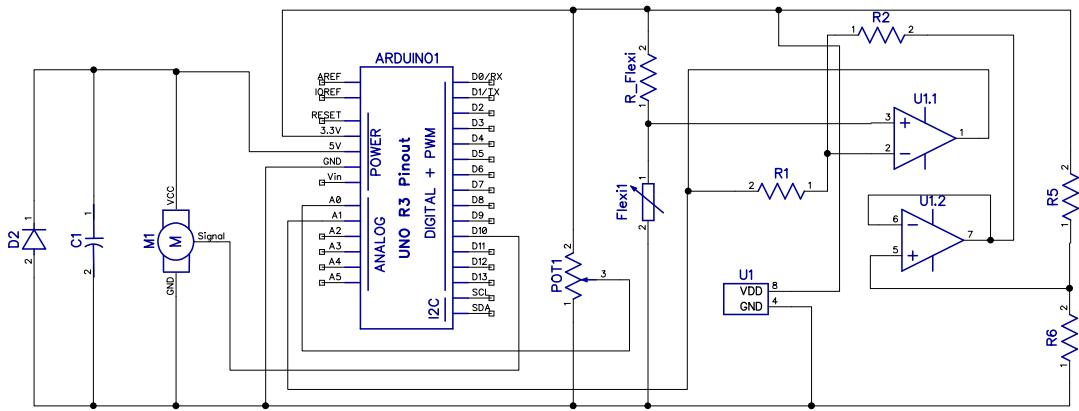
Pravá strana už bola rôzna. Pri verzí TugShield_R2A tu bola súčiastka TLC2274, ktorá obsahovala štyri operačné zosilňovače rail-to-rail a schéma sa zapojila podľa návrhu (Obr. 3.9). Vo verzí TugShield_R2B sa nachádzal TLC2272, ktorý mal len dva opampy a celá schéma bola oveľa jednoduchšia (Obr. 3.10).

Návrh elektrickej schémy sa prekonvertoval do dosky plošných spojov do prostredia PCB Layout. V tomto kroku sa riešila otázka uloženia súčiastok na dosku tak, aby si navzájom neprekážali a aby boli logicky usporiadane. Muselo sa počítať aj s komponentami z 3D tlačiarne, ktoré slúžili na uchytenie serva a flexi snímača a tiež zaberali určité miesto.

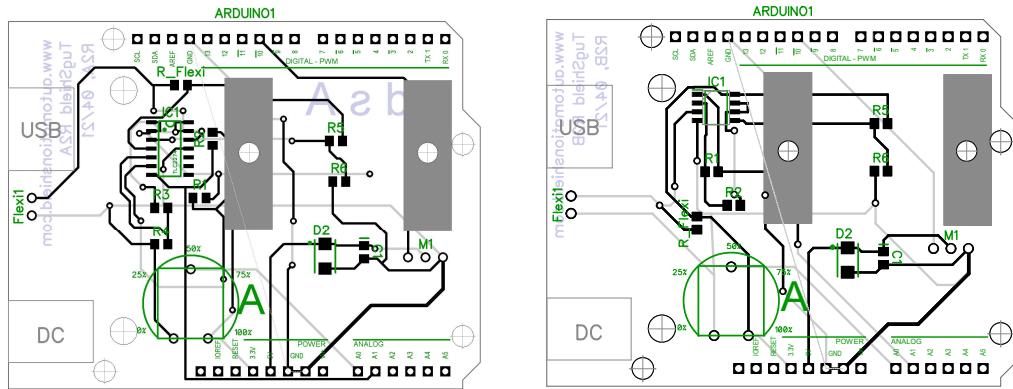
Pozícia flexi snímača bola jednoznačná – ľavý kraj PCB dosky, ďalej sa umiestnilo aj servo na opačný koniec a medzi servomotor a senzor sa položil operačný zosilňovač. Okolo neho sa poskladali rezistory, ktoré s ním priamo súviseli a ostatné, ktoré dotvárali obvod sa zasunuli pod držiak na servo, ktorý bol navrhnutý ako mostík s dvomi stojnami.

Na dosku sa umiestnila grafika projektu a spustil sa Autoroute. Táto funkcia uľahčuje užívateľovi tvorbu pájkovacích ciest medzi jednotlivými komponentami, ktoré boli doteraz spojené len takzvanými “ratlines”, ktoré len naznačujú spojenia podľa predtým navrhnutej elektrickej schémy.

Samozrejme po autotrace si vyžadovala doska ešte menšie úpravy ako nastavenie hrúbky niektorých ciest či prepojenie všetkých pinov GND. Takýto návrh sa poslal do výroby.



Obr. 3.10: Schéma zapojenia TugShield_R2B.

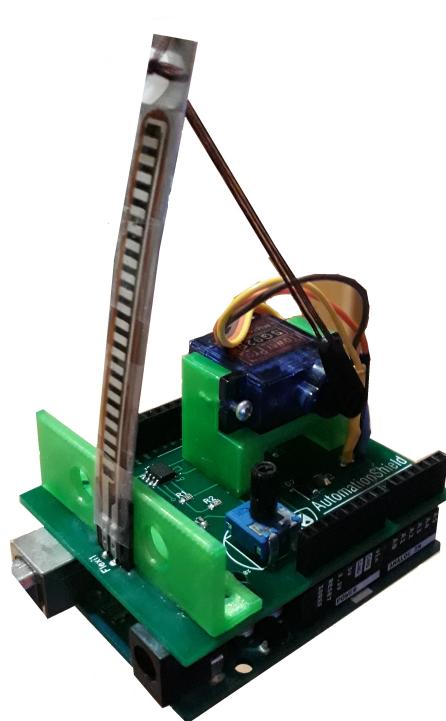


Obr. 3.11: PCB dosky R2A a R2B.

Je dôležité spomenúť, že sice má TugShield komplikovanú schému s množstvom komponentov, ale celková cena nie je vysoká. V Tab. 3.1 je možné vidieť ceny jednotlivých súčiastok a celkovú cenu. Pre užívateľa poskytuje veľký priestor na realizáciu projektov /nápadov za veľmi malú sumu.

Tabuľka 3.1: Cena a informácie o použitých komponentov

Komponent	Výrobca	R2A	R2B
Operačný zosilňovač TLC2272	Texas Instruments	-	1,50€
Operačný zosilňovač TLC2274	Texas Instruments	1,83€	-
Usmerňovač	Vishay Semiconductors	0,41€	0,41€
Rezistor	ROHM Semiconductor	0,63€	0,45€
Flexi snímač	SparkFun	7,94€	7,94€
Microservo	Adafruit	5,20€	5,20€
Kondenzátor	AVX	0,58€	0,58€
Potenciometer	ACP	0,12€	0,12€
PCB doska	PCBWay	0,50€	0,50€
-	Celková cena	15,68€	0,41€



Obr. 3.12: Hotový TugShield R2B.

4 Programátorské rozhranie pre TugShield_R2

TugShiled_R2 nie je komplikované zariadenie z hľadiska fungovania, ale za to poskytuje užívateľom veľa možností pri tvorbe projektov. Jeho základný princíp spočíva v otáčaní ramena mikroserva, ktorý následne ohýba nosník s flexi snímačom. Teda vstup tvorí uhol a výstup primárne reprezentuje napätie. Na výstupe to ale nie je jediná možnosť a pre využitie v ďalších programoch je nevýhodné pracovať s voltami ani s ADC úrovňami, ktoré poskytuje ArduinoIDE. Oveľa prehľadnejšia práca pri riadení je s výstupnými hodnotami v podobe percent.

Z prvého pohľadu sa zdá byť ovládanie tohto shieldu ako elementárny problém, ale časom by užívateľ zistil, že len príkaz na nastavenie uhla ramena by tvoril kód dlhý desať riadkov. Rovnako ako kód na čítanie výstupného signálu z flexi snímača, ktorý je dlhý len približne päť riadkov.

Pre všetky vyššie uvedené dôvody bolo potrebné vytvoriť aplikačné programátorské rozhranie (API), ktoré by zjednodušovalo prístup k TugShieldu a vytvorilo prostredie piateľskejšie pre programátora. Navrhlo a otestovalo sa API pre 3 platformy:

- Arduino IDE
- MATLAB
- Simulink

kde v každej tvorili základ prostredia štyri metódy pokrývajúce základné funkcie riadenia shieldu.

4.1 Arduino IDE

Ako prvé vývojové prostredie pre API sa prirodzene zvolilo Arduino IDE, ktoré je primárne určené na ovládanie Arduina. Softvér využíva modifikovaný jazyk C/C++ a preto aj knižnica určená na implementáciu sa musela napísat v jazyku C. Celé API sa zahrnulo do triedy s názvom TugShield do knižnice `TugShield.h`.

4.1.1 Knižnica TugShield

V hlavičke súboru sa nachádza opatrenie na zabránenie viacnásobného deklarovania knižnice `TugShield.h`.

```
#ifndef TUGSHIELD_H_
#define TUGSHIELD_H_
```

Hned za tým prichádza na rad zahrnutie ostatných knižníc, ktoré TugShield využíva, ako napríklad základná knižnica **AutomationShield**, knižnica pre servomotor – **Servo.h** či knižnica vzorkovania – **Sampling.h**.

```
#include "Arduino.h"
#include "AutomationShield.h"
#include "Servo.h"
#include "Sampling.h"
```

Pre definovanie konštant sa použil príkaz define, ktorý nahradza premennú danou hodnotou. Tento štýl je vhodný použiť na premenné ako čísla pinov, či pevne stanovené konštanty, ktoré sa počas používania rozhrania nebudú meniť.

```
#define TUG_YPIN 1
#define TUG_UPIN 10
#define K_MIN 675
#define K_MAX 0
#define DELAY_VALUE 500
#define SERVO_MAX_FLEX 0
#define SERVO_MIN_FLEX 180
#define DEFAULT_MIN 0.0
#define DEFAULT_MAX 675.0
```

V porovnaní s definovanými konštantami pre TugShield_R1 sa zmenili iba prednastavené hodnoty **DEFAULT_MIN** a **DEFAULT_MAX** z toho dôvodu, že druhá verzia TugShieldu pracuje na 3,3 V logike teda maximum, ktoré signál môže dosiahnuť je 675 ADC úrovní.

Trieda **TugShieldClass** zastrešuje štyri metódy, voľne dostupné pre užívateľa a štyri globálne premenné, ku ktorým užívateľ nemá prístup.

```
class TugShieldClass
{ public:
    void begin();
    void calibration();
    void actuatorWrite(int servo_angle);
    float sensorRead();
private:
    bool _wasCalibrated;
    int _sensorRead ;
    int k_maximal;
    int k_minimal;};
```

Na záver sa vytvoril objekt **TugShield** triedy **TugShieldClass**.

```
extern TugShieldClass TugShield;
```

4.1.2 Metódy

Metóda “Begin”

Na začiatku každého programu je potrebné pre fungovanie TugShieldu inicializovať jeho hardvér. Túto funkciu zabezpečuje metóda s názvom `Begin`, kde sa nastaví pin servomotora `servo.attach(TUG_UPIN)`, jeho rameno sa otočí do začiatočnej polohy 0° `servo.write(SERVO_MIN_FLEX)` a zadá sa príkaz na nastavenie externej analógovej referencie `analogReference(INTERNAL)`.

```
void TugShieldClass::begin(){
    servo.attach(TUG_UPIN);
    servo.write(SERVO_MIN_FLEX);
    analogReference(INTERNAL);}
```

Metóda “Calibration”

Metóda `Calibration` je druhá v poradí logického postupu tvorenia programov. Cieľom tejto metódy je zistiť maximálne a minimálne hodnoty výstupu flexi senzoru pri ohybe ramena serva 0° a 180° .

Meranie sa vykonáva s iteráciou $11 \times$ a postup je nasledovný: Na monitor sériového portu sa vypíše oznam pre užívateľa, že sa začalo kalibrovať. Nasleduje nastavenie ramena do polohy 0° a spustí sa cyklus. V cykle sa vždy zoberie aktuálna hodnota zo snímača a porovnáva sa z predchádzajúcou. V prípade, že sa jedná o minimum, cyklus si do premennej `k_minimal` uschováva najnižšiu hodnotu.

V druhej časti metódy sa celý proces zopakuje pre polohu 180° . Do reálnej premennej `k_maximal` sa zapamätáva maximálna nameraná hodnota z flexi snímača. Po zbehnutí obidvoch cyklov sa flagu s názvom `_wasCalibrated` priradí hodnota `true`, na základe ktorej sa neskôr mapuje výstupný signál. Ako posledné sa oznámi užívateľovi, že kalibrácia prebehla úspešne.

```
void TugShieldClass::calibration(){
    Serial.println("Begin calibration");
    servo.write(SERVO_MIN_FLEX);
    delay(DELAY_VALUE);
    for (int ii=0; ii <= 10; ii++) {
        _sensorRead=analogRead(TUG_YPIN);
        if (ii == 00){
            if(_sensorRead<K_MIN){
                k_minimal=_sensorRead;
                delay(DELAY_VALUE);
            }
        } else{
            if(_sensorRead<k_minimal)
                k_minimal=_sensorRead;
            delay(DELAY_VALUE);
        }
    }
    _wasCalibrated = true;
    Serial.println("TugShield calibrated.");}
```

Metóda “ActuatorWrite”

Pre možnosť nastavenia uhla ramena servomotora sa vytvorila metóda `ActuatorWrite`. Metóda využíva príkazy z knižnice Servo.h a na začiatku spolu s príkazom musí užívateľ uviesť aj hodnotu žiadaného uhla.

Ked'že servo je na TugShielde_R2 otočené opačnou stranou k flexi snímaču, rovnako ako to bolo aj na prvej verzií, musí sa zadaný uhol prepočítať na reálnu hodnotu uhla. Rameno má dráhu polkruhu s rozpätím od 0-180° a preto sa reálna hodnota uhlu získava tak, že sa uhol zadaný užívateľom odpočíta od maxima. Na záver sa táto vypočítaná hodnota zapíše príkazom `servo.write(real_servo_angle)` do servo motora.

```
void TugShieldClass::actuatorWrite(int servo_angle){  
    int real_servo_angle = 180 - servo_angle;  
    servo.write(real_servo_angle);}
```

Metóda “SensorRead”

Poslednou vytvorenou metódou je metóda s názvom `SensorRead`, ktorá je v porovnaní s ostatnými najkomplikovanejšia. Na začiatku sa do premennej `_sensorRead` načíta aktuálna hodnota zo snímača `analogRead(TUG_YPIN)`, ktorý je pripojený na pin A1.

Podmienkou sa zistí, či bol TugShield pred tým nakalibrovaný metódou `Calibration` a od tejto podmienky závisí, ako sa bude mapovať signál z flexi snímača. Ak bola vykonaná kalibrácia, tak sa na premapovanie použijú hodnoty `k_minimal`, `k_maximal`. Ak TugShiled neboli nakalibrovaný, použijú sa defaultné hodnoty 0 a 675.

Premapovanie signálu transformuje singál z ADC úrovni na percentá, s ktorými sa dá jednoduchšie pracovať pri riadení a v ďalších príkladoch.

```
float TugShieldClass::sensorRead(){  
    _sensorRead = analogRead(TUG_YPIN);  
    float _sensorValue;  
    if(_wasCalibrated == true){  
        _sensorValue = AutomationShield.mapFloat((  
            float)_sensorRead,(float)k_minimal,(  
            float)k_maximal, 0.00, 100.00); }  
    else{  
        _sensorValue = AutomationShield.mapFloat((  
            float)_sensorRead, DEFAULT_MIN,  
            DEFAULT_MAX, 0.00, 100.00); }  
    return _sensorValue; }
```

4.2 MATLAB

MATLAB od MathWorks je vysoko výkonný softvér pre technické výpočty. Integruje výpočty, vizualizáciu a programovanie do ľahko použiteľného prostredia. Jeho hlavnými výhodami v porovnaní s ArduinomIDE je, že obsahuje mnoho jednoduchých príkazov, ktoré zahŕňajú inak komplexné funkcie a programátorom uľahčuje prácu [43].

Ďalšou veľkou výhodou je zobrazovanie dát, grafov, ktoré je mimoriadne prehľadné, rovnako ako práca s nimi. Na rozdiel od ArduinaIDE sa dá priamo na graf doplniť legenda, názov, pomenovanie osí a ďalšie.

Všetky vyššie uvedené výhody sa dajú aplikovať aj pri tvorbe rozhrania pre TugShield_R2. V MATLABe sa vytvorili štyri podobné funkcie ako v ArduinoIDE metódy až na to, že sa v kóde museli použiť príkazy, ktoré umožňujú spoluprácu MATLABu s platformou Arduino.

Ako prvé sa vytvorila trieda `TugShield` príkazom `classdef TugShield < handle`

```
classdef TugShield < handle
```

V ďalšom kroku sa do kategórie public – dostupné užívateľovi – zaradili objekty `arduino` a `servo` a globálna premenná `was_calibrated`. Premenná slúži ako flag v metóde `Calibration` a je typu boolean.

```
properties(Access = public)
    arduino;
    servo;
    was_calibrated = false;
    k_minimal;
    k_maximal;
end
```

Do skupiny konštánt sa zaradili rovnaké premenné ako v knižnici `TugShield.h`, ale hodnoty niektorých sa zmenili. Napríklad uhol ramena servomotora sa nastavuje v rozpätí od 0 po 1.

```
properties(Constant)
    TUG_YPIN = 'A1';
    TUG_UPIN = 'D10';
    K_MIN = 675;
    K_MAX = 0;
    DELAY_VALUE = 0.5;
    SERVO_MAX_FLEX = 0;
    SERVO_MIN_FLEX = 1;
    DEFAULT_MIN = 500.0;
    DEFAULT_MAX = 1000.0;
end
```

Funkcia “Begin”

Funkcia `begin` v MATLABe vyžaduje od užívateľa, aby zadal názov objektu, číslo portu a druh dosky Arduino. Načíta sa Arduino knižnica `listOfLibraries=listArduinoLibraries()`, servo knižnica a skontroluje sa či je naozaj prítomná.

Zadefinujú sa parametre objektu pre `arduino` z údajov, ktoré zadal do funkcie užívateľ `TugShieldObject.arduino = arduino(aPort, aBoard, 'Libraries', 'Servo')` a rovnako sa inicializuje aj servo objekt, ktorému sa priradí pin vstupného signálu. Nastaví sa začiatocná poloha ramena serva a po úspešnom zvládnutí všetkých krokov sa používateľovi vypíše oznam, že `TugShield` bol inicializovaný.

```

function begin(TugShieldObject , aPort , aBoard)
listOfLibraries = listArduinoLibraries();
libraryIsPresent = sum(ismember(listOfLibraries , 'Servo'))
);
if ~libraryIsPresent
    error('Servo library was not detected by MATLAB !')
end
TugShieldObject.arduino = arduino(aPort , aBoard , ,
    Libraries' , 'Servo');
TugShieldObject.servo = servo(TugShieldObject.arduino ,
    TugShieldObject.TUG_UPIN);
writePosition(TugShieldObject.servo , TugShieldObject.
    SERVO_MIN_FLEX);
disp('TugShield initialized .')
end

```

Funkcia “Calibration”

Hodnoty premenných `k_minimal` a `k_maximal`, ktoré predstavujú najmenšiu a najväčšiu nameranú hodnotu, sa zisťujú vo funkcií `calibration`. Realizuje sa meranie v dvoch pozícia ramena serva, kde sa v každej pozícii 10 krát odčíta hodnota z flexi snímača a porovná s predošlou nameranou. Na záver sa globálnej premennej `was_calibrated` priradí hodnota `true`, ak bola kalibrácia vykonaná správne.

```

function calibrate(TugShieldObject)
    disp('TugShield calibrated .')
    writePosition(TugShieldObject.servo , TugShieldObject.
        SERVO_MIN_FLEX);
    for i=1:10
        sensorRead=readVoltage(TugShieldObject.arduino ,
            TugShieldObject.TUG_YPIN)
        if(sensorRead<TugShieldObject.K_MIN)
            k_minimal=sensorRead;
        end
    end
    was_calibrated = true;
    disp('TugShield calibrated .')
end

```

Funkcia “ActuatorWrite”

Metóda v ArduinoIDE a funkcia v matlabe `Actuatorwrite` majú presne rovnaký postup a svojím spôsobom aj rovnaké príkazy. Jediná zmena nastala v tom, že MATLAB udáva polohu ramena v rozpätí od 0 do 1 a teda pri prepočítavaní reálneho uhla sa hodnota zadaná užívateľom odčítava od 1 – maximálnej krajnej polohy serva.

```

function actuatorWrite(TugShieldObject , angle)

```

```

    right_angle = 1 - angle;
    writePosition(TugShieldObject.servo,right_angle);
end

```

Funkcia “SensorRead”

Poslednou potrebnou funkciou pre rozhranie v MATLABe je funkcia `sensorRead`. Má za úlohu načítať signál z flexi snímača a premapovať ho na percentá.

Na výber sú dve možnosti premapovania podľa toho či bola vykonaná kalibrácia. Rozhoduje o tom premenná `was_calibrated`, do ktorej je priradená hodnota true, ak bol TugShield kalibrovaný. Inak sa použijú defaultné hodnoty.

```

function y = sensorRead(TugShieldObject)
    sensorRead = readVoltage(TugShieldObject.arduino,
        TugShieldObject.TUG_YPIN);
    if was_calibrated == true
        y = map(sensorRead, TugShieldObject.k_minimal,
            TugShieldObject.k_maximal, 0.00,100.00);
    else
        y = map(sensorRead, TugShieldObject.DEFAULT_MIN,
            TugShieldObject.DEFAULT_MAX, 0.00,100.00);
    end
end

```

4.3 Simulink

Spoločnosť MathWorks vyvinula aj nadstavbu MATLABu –Simulink, pre simuláciu a modelovanie dynamických systémov. Je to grafické programovacie prostredie, v ktorom sa príkazy a funkcie skrývajú za blokmi.

Simulink funguje na trochu odlišnom princípe ako bežné programovacie jazyky a preto aj TugShield API nebude rovnaké ako pri MATLABe a Arduino IDE. Niektoré funkcie ako napríklad Calibration alebo Begin by nemali zmysel alebo by ich nebolo možné použiť, pretože v Simulinku sa nedajú určiť špecifické podmienky pre prvú iteráciu v cykle.

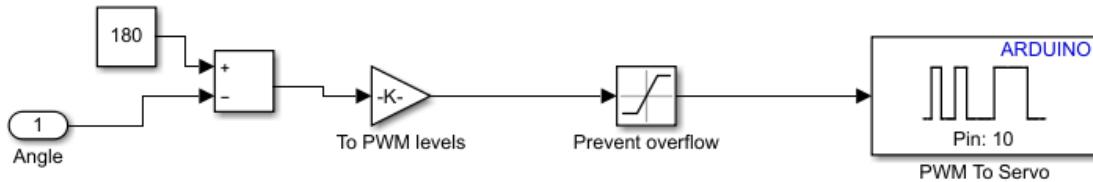
Na základe toho sa vytvorili teda tri bloky, ktoré pokrývajú základné činnosti TugShieldu a inicializácia hardvéru a kalibrácia sa implementovali priamo do nich.

Blok “ActuatorWrite”

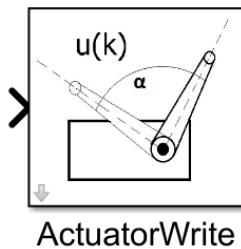
Blok ActuatorWrite slúži na zapísanie uhla do ramena serva. Vyžiada sa hodnota uhla od užívateľa, ktorá sa potom pripojí do bloku Add so znamienkom mínus a odčíta sa on konštanty 180° . Výsledná hodnota sa transformuje na PWM signál, ktorý následne prejde cez blok Prevent overflow, ktorý udáva saturačné hodnoty a signál sa pošle na desiaty pin, na ktorom je umiestnené servo Obr. 4.1.

Celý reťazec blokov sa zahrnie ako subsystém do bloku s názvom ActuatorWrite. Pre tento blok sa vytvorí maska, ktorej sa priradí ikona charakterizujúca funkciu bloku Obr. 4.2.

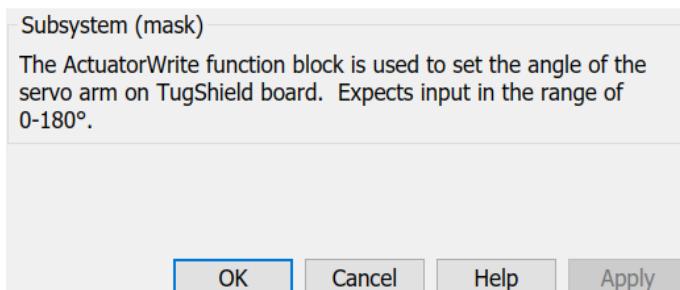
Pre tento blok sa po dvojkliknutí zobrazí maska, kde je stručný popis Obr.4.3.



Obr. 4.1: Reťazec blokov vytvárajúci ActuatorWrite.



Obr. 4.2: Blok ActuatorWrite.



Obr. 4.3: Maska bloku ActuatorWrite.

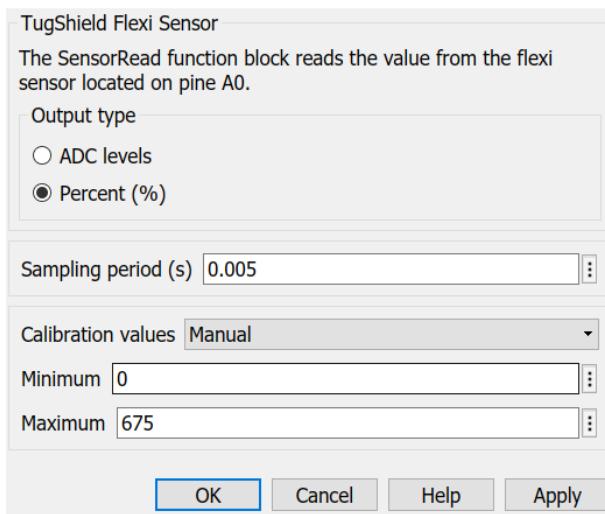
Blok "SensorRead"

SensorRead slúži na odčítanie signálu z pinu A1 a užívateľovi ponúka možnosť, v akom formáte sa má tento signál spracovať. Výstup može byť reprezentovaný v ADC úrovniach alebo v percentách. Od toho sa odvíjajú aj ďalšie nastavenia v maske, ktoré si užívateľ môže navoliť. Ak zvolí možnosť percentá, môže si vybrať, akým spôsobom budú zvolené krajiné hodnoty na premapovane singálu. Buď automatické, ktoré sa zistia kalibráciou alebo manuálne, ktoré zadá programátor Obr. 4.4.

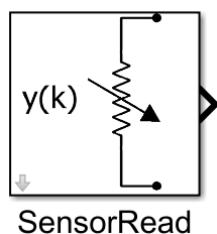
Všetky interaktívne prvky sú niečím podmienené, čím sa predchádza nefunkčnosti príkladov, v ktorých by bol tento blok použitý. Napríklad maximum a minimum musia byť v danom intervale a výber medzi manuálnymi alebo prednastavenými hodnotami sa dá iba v prípade, ak sú zvolené percentá ako výstup.

Vo vnútri bloku sa nachádza blok s implementovanou MATLAB funkciou outputReading Obr. 4.6, ktorej vstupné hodnoty sú: OutRadio, MinVal, MaxVal a blok čítajúci signál z pinu 1. Premenná OutRadio sa nastaví, keď si užívateľ navolí spôsob výstupu a následne sa podľa nej zvolí z dvoch variánt mapovania Obr. 4.6.

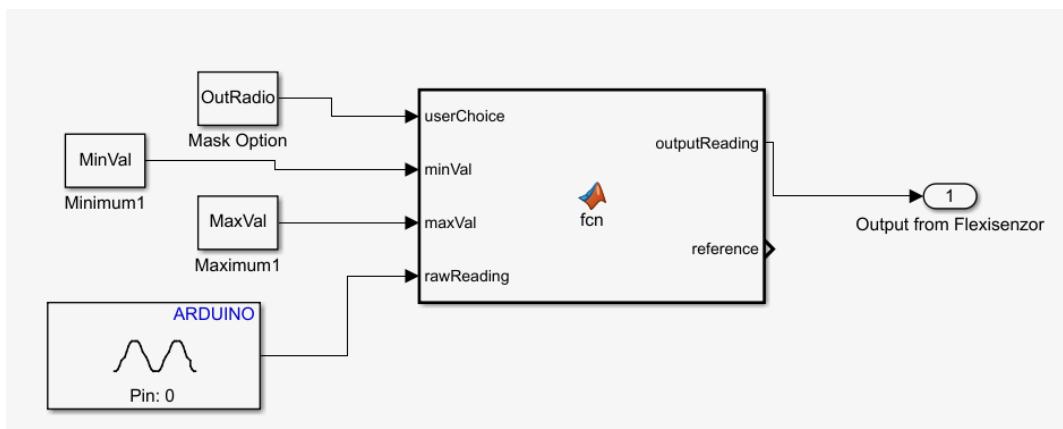
MinVal a MaxVal sa musia vytvárať čisto v matlab kóde ešte pred použitím tohto bloku v cykle v simulinku. Je to nevýhoda, ktorá sa spomínala už aj vyššie. Simulink nedokáže



Obr. 4.4: Interaktívna maska bloku SensorRead



Obr. 4.5: Blok SensorRead.



Obr. 4.6: Vnútro bloku SensorRead.

oddeliť jednotlivé iterácie cyklu a preto sa kalibračné hodnoty musia získať predom a potom len použiť v hlavnom programe Simulinku.

```
function [outputReading ,reference] = fcn(userChoice ,
minVal , maxVal , rawReading)
persistent ref ;
if userChoice == 1
    outputReading = rawReading ;
```

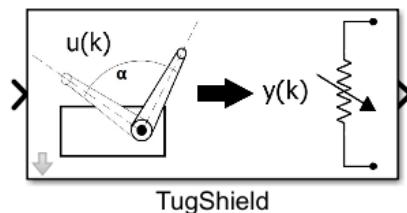
```

else
    if isempty(ref)
        ref = 1;
        TugShield.actuatorWrite(0);
        pause(1);
        minVal = TugShield.sensorRead();
        pause(1);
        TugShield.actuatorWrite(180);
        pause(1);
        maxVal = TugShield.sensorRead();
    end
    reference = ref;
    outputReading = interp1([minVal,maxVal]
                           ,[0.00,100.00],sensorRead);
end

```

Výstupom z toho bloku je premapovaný signál z flexi snímača.

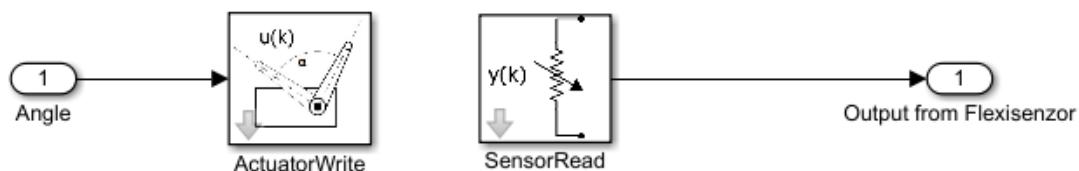
Blok “TugShield”



Obr. 4.7: TugShield blok.

Posledným vytvoreným blokom je TugShield Obr. 4.7, ktorý má za úlohu spojiť ActuatorWrite a SensorRead. Vytvára tak plynulé nastavovanie uhla servomotora a následné odčítavanie signálu z flexi snímača.

Tento blok bolo potrebné vytvoriť hlavne z toho dôvodu, že riadiaci cyklus by sa ľahšie skonšturoval, keby tieto dva bloky boli oddelené, pretože sa nedajú prirodzene spojiť za seba.

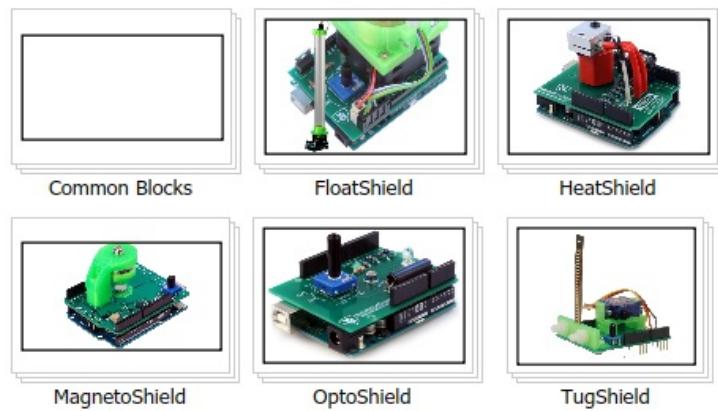


Obr. 4.8: Spojenie blokov ActuatorWrite a SensorRead vo vnútri bloku

Zahrnutie do knižnice AutomationShield

Simulink knižnica AutomationShield združuje vytvorené bloky zo všetkých projektov a teda aj bloky TugShieldu musia byť súčasťou tejto knižnice.

Knižnica ponúka komfotné menu, kde sa dá navoliť skupina blokov pomocou kniknutia na obrázky, ktoré zobrazujú shieldy. Vytvoril sa nový prázdny subsystém, nastavila sa ako ikona fotka TugShieldu Obr. 4.9 a do callbacku na nastavil link na knižnicu TugShield, ktorá sa zobrazovala po kliknutí.



Obr. 4.9: Simulink knižnica AutomationShield so zahrnutým TugShieldom

5 Demonštračné príklady

V predchádzajúcej kapitole bolo podrobne popísané vytvorenie metód v rôznych prostrediach na ovládanie TugShieldu. Pre overenie či boli metódy vytvorené správne a či pokrývajú všetky základné funkcie shieldu sa vytvorili demonštračné príklady.

Príklady majú za úlohu užívateľovi ukázať programátorské možnosti, využitie vytvorených metód a predstaviť shield z pohľadu fyzikálneho princípu. Zhotovilo sa niekoľko príkladov v každom prostredí, ktoré testujú správnosť kalibrácie, identifikujú sústavu alebo riadia systém.

5.1 Matematicko-fyzikálna analýza

Pred riešením príkladov je veľmi dobré sa zamyslieť nad fyzikálnym princípom, na ktorom TugShield pracuje a pokúsiť sa ho analyzovať. Pri regulátoroch ako PID, ktoré nevyužívajú pri riadení model je viac menej zbytočná, ale pre LQ a MPC riadenia sa bez modelu nedá pohnúť. Výhody a neýhody jednolivých regulátorov sa dajú zhrnúť v krátkosti. LQ a MPC riadenia sú oveľa presnejšie, teda vedia lepšie zregulovať systém, ale sú mimoriadne pamäťovo náročné a tým pádom v praxi aj drahšie. Na rozdiel od PID regulátora, ktorý je v praxi najroširenejší, pretože jeho cena je sice úmerná účinnosti riadenia, ale poväčšine je to dostačujúce.

Ohyb je definovaný ako uhol alebo vzdialenosť, do ktorého je konštrukčný prvok posunutý pod zatažením, teda v dôsledku deformácie. Veľkosť deformácie závisí od rôznych faktorov ako napríklad veľkosť a smer pôsobiacej sily, od materiálu či geometrie nosníka.

Pod geometriou nosníka sa chápe hlavne jeho tvar a prierez. V počiatočnom stave bez záťaže môže byť celkový tvar rovný alebo zakrivený, čo sa týka prierezu, veľkosť deformácie ovplyvňuje tvar priezera, ale aj to či je tento tvar po celej dĺžke konštantý. Materiál zohráva významnú úlohu tiež, každý materiál má inú tuhost. Niektoré nosníky nemusia byť ani materiálovou homogénne, ale kompozitné (zložené z rôznych materiálov).

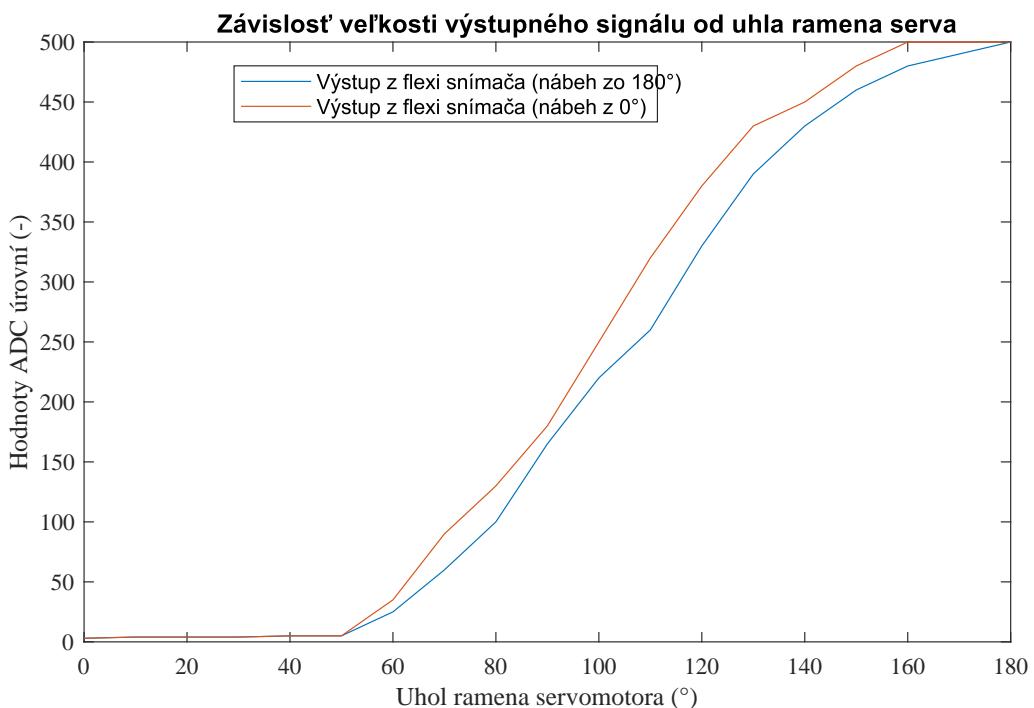
Všetky tieto faktory ovplyvňujú výslednú deformáciu a niektoré z nich stážujú analýzu, ale veľa technických aplikácií zahŕňa prípady, ktoré nie sú také zložité. Analýza je zjednodušená ak [42]:

- nosník je rovný po celej dĺžke,
- nosník vykonáva iba lineárnu elastickú deformáciu,
- nosník je štíhly (pomer dĺžky a výšky je väčší ako 10:1),
- analyzuje sa malá výchylka (maximálna výchylka menšia ako 1/10 rozpätia).

V prípade TugShieldu je nosník vyrobený z kalenej ocele a materiál je v každej časti homogénny. To isté platí aj o jeho celkovom tvare, ktorý sa nemení cez celú dĺžku. Nosník sa môže považovať za štíhly, pretože je dlhý 11 cm a široký len 1 cm. Ako polotovoar na výrobu bol použitý plech a kedže je nosník k servomotoru pripojený v krajnej časti presne v strede, nie je to tvar, ktorý by robil analýzu komplikovanejšiu.

Čo ale jednoznačne zvlaží vytváranie modelu je priebeh celej deformácie. V prípade TugShieldu sa nebude riadiť len časť s malou výchylkou, v ktorej by sa mohla považovať deformácia za lineárnu. Ani z celkového hladiska sa nemôže považovať deformácia za lineárnu, pretože ohyb spôsobuje rameno idúce po kružnici.

Ako prvé sa teda vykonala statická analýza celého systému, ktorá mala určiť linearitu TugShieldu. Na TugShielde sa postupne nastavoval uhol ramena servomotoru od 0° - 180° s krokom 10° . Odčítavali sa dve hodnoty a to uhol ohnutia nosníka a ADC úroveň, ktorá bola ako výstup z flexi snímača.



Obr. 5.1: Závislosť výstupného signálu od uhla ramena serva.

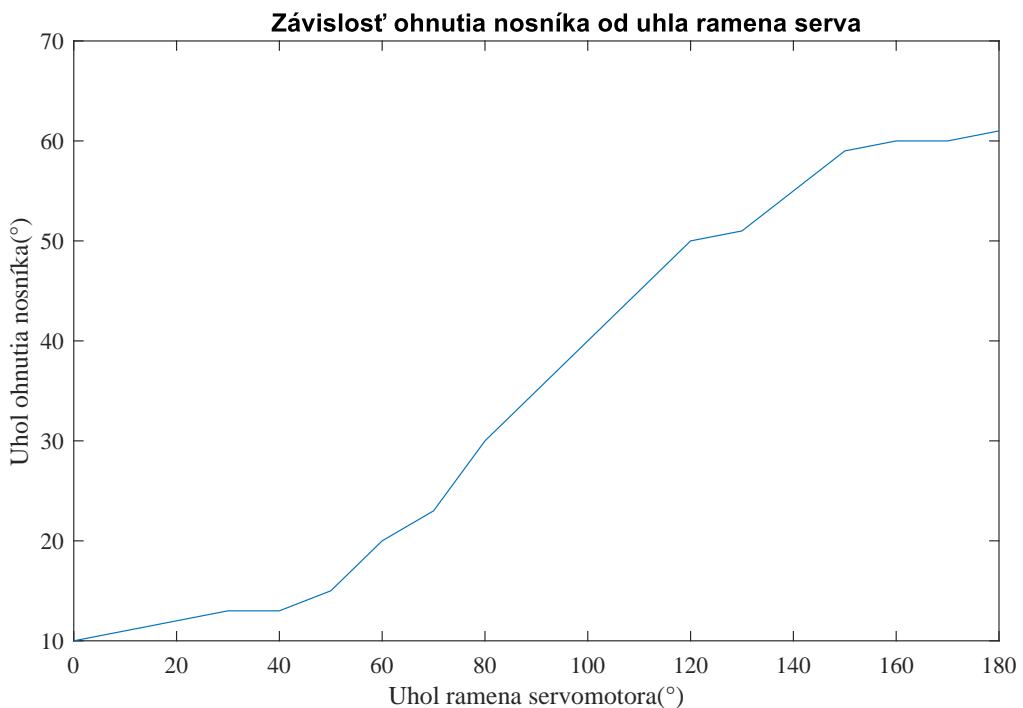
Kedže snímač lepšie reaguje na väčšie zmeny, pri každej zmene uhlia o ďalší krok sa najprv nastavila počiatočná poloha ramena a až potom sa nastavil nasledujúci uhol v kroku, kde sa odčítavala výstuná hodnota. Meranie sa vykonalo dvakrát pričom raz sa nabiehalo z počiatočnej polohy 0° a druhýkrát z polohy 180° .

Výsledkom merania sú dva grafy. Z grafu, ktorý zobrazuje závislosť výstupného signálu od natočenia ramena, sa dá jasne vidieť, v akých častiach je možné sýstém linearizovať. Od približne 50° po 130° je signál priamoúmerný veľkosti uhlia ramena a v tomto rozpätí sa bude systém jednoducho analyzovať.

Od 0° - 50° a 130° - 180° sa systém nespráva lineárne a tak, ako by sa predpokladalo. Je to spôsobené tým, ako už vyššie bolo spomenuté, že rameno nerobí priamočiary pohyb

a teda sila, ktorá pôsobí na nosník nie je vždy rovnobežná s doskou TugShieldu. Uhol pôsobenia sily sa mení každým stupňom otočenia ramena.

Na druhom grafe je veľmi podobná charakteristika systému s tým rozdielom, že na y-ovej osi je vynesený uhol ohnutia nosníka. Z tohto grafu je tiež jasne vidieť časť, ktorú je možné považovať za lineárnu od 50° - 150° .



Obr. 5.2: Závislosť ohnutia nosníka od uhla ramena serva.

Toto meranie sa vykonalo z toho dôvodu, že snímač má istú citlivosť a na malé zmeny nereaguje, pričom ohnutie nosníka sa zmenilo aj o niekoľko stupňou. Rozdiel, ktorý teda spôsobuje necitlivosť snímača je možné vidieť hlavne v časti od 0° - 50° uhla ramena, kde výchylka nosníka je približne od 10° - 15° a snímač na túto zmenu takmer vôbec nereaguje.

Ale keďže riadit a merať sa nebude samotná výchylka nosníka, ale to, ako na ňu reaguje snímač, tým pádom je smerodajný pre TugShield prvý graf.

V lineárnej oblasti môžeme teda rovnicu popisujúcu ohyb nosníka aproximovať ako [42]:

$$d^2w(x)/dx^2 = M(x)/E(x)Jz(x) \quad (5.1)$$

kde druhá derivácia uhla vychýlenia w vzhľadom na x sa interpretuje ako zakrivenie, E je Youngov modul, Jz je plošný moment zotrvačnosti prierezu a M je vnútorný ohybový moment nosníka.

Ak je nosník homogénny a pôsobí naň rozložené zaťaženie q , je možné vyššie uvedený výraz napísť ako [42]:

$$E(x)Jz(x)d^4w(x)/dx^4 = q(x) \quad (5.2)$$

Nosník na TugShielde sa definuje ako typ konzolového nosníka, ktorý ma jeden koniec votknutý a odoberá dva stupne voľnosti a druhý koniec má voľný. Rov. (5.2) je teda možné modifikovať pre konzolové nosníky na tvar [27]:

$$\delta = FL^3/3EJz \quad (5.3)$$

a

$$\theta = FL^2/2EJz \quad (5.4)$$

Sila pôsobí na konci nosníka a preto vo Rov. (5.3) a Rov. (5.4) sa $E(x)$ a $Jz(x)$ mení na E a Jz , čo znamená, že za x sa dosadila dĺžka nosníka a uvažuje sa ďalej rovnica len v zjednodušenom zápise. Výsledkom sú teda vzťahy vyjadrujúce pružné vychýlenie δ a uhol vychýlenia θ . Tieto dva vzťahy by mohli byť základom pre vytvorenie modelu v budúnosti.

5.1.1 Experimentálna analýza

Analýza systému a vytvorenie modelu sa dá urobiť dvoma spôsobmi. Jeden koncept bol načrtnutý v predošej podkapitole a na druhý sa zameria aktuálna kapitola.

Po experimentálnej analýze, teda black-box modeli sa siahá v tom prípade, ak existuje množstvo vstupno-výstupných dát ale nedá sa presne matematicky vyjadriť, čo sa s dátami deje vo vnútri systému. V takomto prípade sa spracujú dáta a aplikujú sa do softvéru alebo nástroja, ktorý vytvorí black-box model.

Jeden z nástrojov, ktorý toto dokáže je MATLAB System Identification Toolbox. Ako vstup sa môžu použiť vstupno-výstupné dáta v časovej doméne a vo frekvenčnej doméne na identifikáciu funkcií prenosu v nepretržitom a diskrétnom čase, procesných modelov a modelov v stavovom priestore [25].

V prvom rade je teda potrebné získať dáta, ktoré sa použijú do System Identification. Vstupné dáta sa dajú získať rôznymi spôsobmi. Najzdlhovejší a najnehodnejší je vytvoriť si ručne maticu vstupov, ktorá by sa použila ako vstupný signál.

Samozrejme sa vygenerovanie vstupných dát dá spraviť oveľa jednoduchšie cez MATLAB skript. Treba však pri tom zohľadniť viacero faktorov. Vstupný signál musí byť dostatočne dynamický, ale len do takej miery, aby naň dokázal TugShield reagovať, pretože aj natáčanie ramena a zaznamenanie signálu z flexi trvá nejaký čas.

```
N = 4800;
minu = 80;
maxu = 150;
B = 1 / 100;
prbsGenerate('prbsU', N, minu, maxu, B)
```

Vytvorili sa dva skripty, ktoré náhodne generovali vstupný signál podľa požiadaviek. Rozdiel medzi skriptami APRBS a PRBS je, že PRBS generuje vstupný signál, ktorý sa v náhodných časových intervaloch skokovo mení z maximálnej na minimálnu, no APRBS využíva celý rozsah a vstupný signál tvoria rôzne úrovne.

Nasledovalo vytvorenie výstupného signálu z flexi snímača. Vygenerované signály APRBS a PRBS sa jednotlivo aplikovali na TugShield ako vstupy so snahou získať odozvu na tento signál.

```

#include "prbsU.h"
void step() {
    if(k>prbsU_length) {
        TugShield.actuatorWrite(0.0);
        while(1);
    } else {
        prbs = pgm_read_word(&prbsU[k]);
        u = PIDAbs.compute(prbs-y,0,180,0,100);
    }
    TugShield.actuatorWrite((int)u);
    y = TugShield.sensorRead();
    Serial.print(y);
    Serial.print(" ");
    Serial.println(u);
    k++;
}

```

Posledný krok pred samotnou analýzou bol transport výstupného signálu z flexi snímača, ktorý bol získaný na platforme Arduino IDE, do MATLABu. Na to sa využila funkcia zaznamenávania a ukladania signálu z Arduino IDE do MATLABu, ktorá sa musela spustiť tesne pred zapnutím programu v Arduino IDE.

```

function result=readExperiment(lngth,port,baud)
s = serial(port);
set(s,'BaudRate',baud);
fopen(s);
result=[];
sampleText = fscanf(s);
for i=1:lngth;
sampleText = fscanf(s);
sampleCell = textscan(sampleText ,'%f%f' , 'Delimiter
',',',' ');
result(i,:)=sampleCell{1};
end
fclose(s);

```

V tomto momente sa už mohlo prejsť na analýzu. Do System Identification Toolboxu sa nahrali vstupno-výstupné dátá vynesené v čase a zvolil sa typ výstupnej analýzy. Typ závisí od toho, pre aké riadenie sa ide model využívať. Pokiaľ by užívateľ chcel vytvoriť PID riadenie na nasimulovanom systéme napríklad v Simulinku, použil by "Transfer Function Models", ktorý by vytvoril prechodovú charakteristiku.

V prípade TugShieldu a ďalšieho použitia modelu sa zvolila možnosť "State Space Model", ktorý vytvorí lineárny časovo invariantný stavový model, nutný pre LQ a MPC regulátory v demonštračných príkladoch.

5.2 Príklady

Pre lepšiu prezentáciu používania vytvorených metód a všeobecného fungovania TugShieldu a možnosti jeho použitia sa vytvorili demonštračné príklady.

Pre všetky softvérové platformy sa vytvorili príklady na rovnakej báze, ale museli byť modifikované, čo znamená, že každý softvér má iný, ale podobný výstup.

5.2.1 Príklad FirstCheck

Tento príklad je predchodom príkladu SelfTest, ale je ešte zjednodušený a poskytuje priame výsledky merania. Bol vytvorený len na otestovanie obidvoch verzií TugShield_R2A a TugShield_R2B a vďaka nemu sa overila funkčnosť.

```
TugShield.actuatorWrite(90);
delay(1000);
y = TugShield.sensorRead();
Serial.println(y);
TugShield.actuatorWrite(160);
delay(1000);
y = TugShield.sensorRead();
Serial.println(y);
TugShield.actuatorWrite(30);
delay(1000);
y = TugShield.sensorRead();
Serial.println(y);
```

Výsledky príkladu ukázali, že TugShield_R2A vôbec nereaguje na zmenu polohy ramena serva, v porovnaní s TugShield_R2B, ktorý reaguje primerane aj keď zosilnenie nie je ideálne, ale shield je funkčný Obr.5.3.

TugShield calibrated.	TugShield calibrated.
0	9
0	84
0	0
0	12
0	87
0	0

Obr. 5.3: Výsledky pre TugShield_R2A a TugShield_R2B z príkladu FirstCheck.

5.2.2 Príklad SelfTest

Po skonštruovaní kompletného hardvéru a vytvorenia TugShield triedy, úlohou prvého testovacieho príkladu bolo overiť funkčnosť metód a vstupného aj výstupného signálu. Príklad bol jednoduchý ale výstižný, nevyužíval žiadne ďalšie knižnice z AutomationShield.

Arduino IDE

Princípom bolo v úvode inicializovať hardvér a nakalibrovať ho použitím dvoch metód `TugShield.begin` a `TugShield.calibration`. Potom sa rameno servomotra na-

stavilo do minimálnej krajnej hodnoty 0° metódou `TugShield.sensorRead` a ďalšou metódou `TugShield.actuatorWrite` sa získal premapovaný signál z flexi snímača.

Premapovaný signál v krajnej minimálnej polohe by sa mal pohybovať okolo hodnoty 0° a preto sa kontrolovalo či sa nachádza pri tejto hodnote s 10% toleranciou do plusu. Rovnaký postup sa zopakoval aj pre maximálnu krajnú polohu 180° .

```
TugShield.begin();
TugShield.calibration();
delay(100);
Serial.println("Testovanie flexi snimaca... poloha serva: 0");
TugShield.actuatorWrite(0);
delay(500);
y = TugShield.sensorRead();
if (y >= 0.0 && y <= 10.0) {
    Serial.println("V poriadku.");
    Serial.println(y);
} else {
    Serial.println("Nevyhovuje.");
    Serial.println(y); }
```

MATLAB

Rovnako sa príklad realizoval aj v MATLABe. Vytvoril sa `TugShield` objekt, inicializoval sa hardvér pomocou `TugShield.begin('COM3', 'UNO')`, kde vstupnými údajmi sú číslo portu a typ dosky, následne sa nakalibroval a vypísal odkaz pre užívateľa, že test prebieha.

Nastavila sa poloha ramena serva 0° a pri tejto polohe sa skontroloval signál z flexi snímača či sa nachádza v intervale $0\text{-}10\%$.

```
TugShield = TugShield;
TugShield.begin('COM3', 'UNO');
TugShield.calibrate();
disp('Testovanie flexi snimaca... poloha serva: 0')
TugShield.actuatorWrite(0);
pause(1);
y = TugShield.sensorRead();
if (y >= 0.0 && y <= 10.0)
    disp("Hodnoty flexi snimaca su v poriadku");
    disp(y);
else
    disp("Hodnoty flexi snimaca nevyhovujuce");
    disp(y);
end
pause(0.5);
```

```

TugShield calibrated.
Flex sensor test...servo position: 0
The values of flex sensor are OK
    0

Flex sensor test...servo position: 180
The values of flex sensor are OK
    96.6667

Test completed
```

```

Obr. 5.4: Výsledom príkladu SelfTest v Matlabe.

### 5.2.3 Príklad Identification

Po overení funkčnosti metód na kalibráciu a inicializovanie hardvéra, ktoré môžeme zaradiť do kategórie statickej kontroly, nasleduje dynamická. Je rozdiel medzi signálom, ktorý je získaný v kľudovom režime a signálom, ktorý sa mení v čase podľa uhla ramena serva. Pre kontrolu, ako sa správa signál z pohľadu, ako dokáže sledovať vstupnú hodnotu, aký je čas reakcie a všetky ďalšie potrebné veci na riadenie, sa vytvoril príklad Identification.

#### Arduino IDE

V prvej časti sa definujú potrebné premenné a konštandy  $T_s$  - perióda vzorkovania,  $U[]$  - pole vstupov,  $T$  - dĺžka trvania sekcie, inicializuje sa hardvér a TugShield sa nakalibruje. Vstupný signál je definovaný ako trajektória preddefinovaných vstupov a tvorí ju desať hodnôt uhla ramena. V tomto príklade sa využívajú aj metódy z knižnice AutomationShield, ktoré pomáhajú určovať periódu vzorkovania či dĺžku trvania jednotlivej sekcie z pola vstupov.

V jadre programu sa nachádza funkcia, ktorá sleduje či prešiel dostatočne dlhý čas na to, aby sa mohla vykonať ďalšia vzorka. Ak flag premenná dovolí tento krok, prechádza sa do druhej hlavnej časti programu, kde sa vykonáva samotné meranie a zapisovanie signálu z pola vstupov. Vzorka sa odoberá každých 10 ms, pričom vstupný signál mení hodnotu len každú sekundu.

Výsledkom príkladu je graf, na ktorom je zobrazený vstupný a výstupný signál.

```

unsigned long Ts = 10;
unsigned long k = 0;
bool enable=false;
int U[]={20, 50, 15, 90, 60, 120, 75, 180, 40, 80};
int T = 1000;

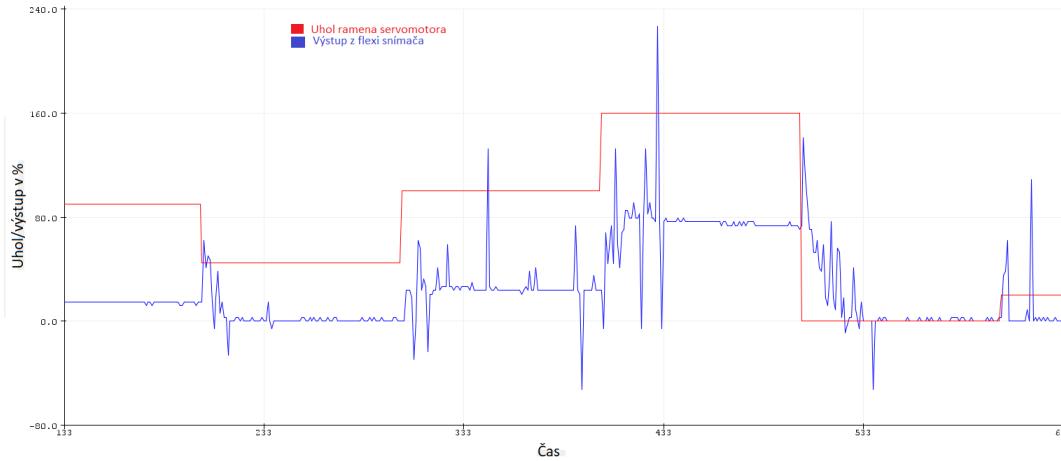
void step(){
if (k % (T*i) == 0){
 u = U[i];
 i++;
}

```

```

TugShield.actuatorWrite(u);
y = TugShield.sensorRead();
Serial.print(y);
Serial.print(" ");
Serial.println(u);
k++; }

```



Obr. 5.5: Výstup príkladu Identification v Arduino IDE.

## MATLAB

V prostredí MATLAB má príklad rovnakú štruktúru. Výhodou MATLABu je zabudovaná funkcia na meranie času **`tic - toc`**, kde **`tic`** znamená začiatok merania a **`toc`** koniec. Funkcia sa dá jednoducho využiť na sledovanie pretečenia času medzi jednotlivými vzorkami.

Ďalším rozdielom je vykreslovanie grafov. V Arduino IDE je možnosť otvorenia sériového monitoru, ktorý automaticky vykreslí, čo sa práve deje, ale s grafom sa nedá manipulovať. V MATLABe sa s grafom dá pracovať, znamená to nastaviť legendu, pomenovanie osí, priblíženie, čo sú dobré nástroje na získanie presnejších informácií.

```

R = [20, 50, 15, 90, 60, 120, 75, 180, 40, 80];
T = 40;
n = length(R);

tic
while (1)
 if (nextStep)
 if (mod(k, T*i) == 1)
 i = i + 1;
 if (i > length(R))
 TugShield.actuatorWrite(0.0);
 break

```

```

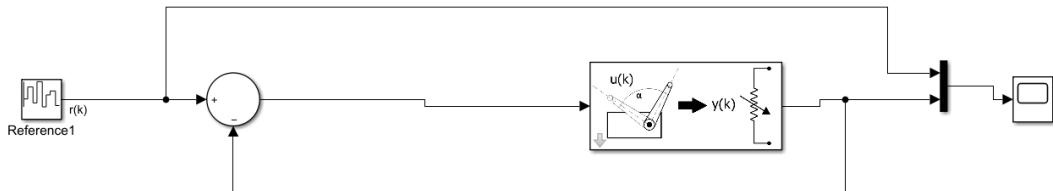
 end
 u = R(i);
 end
 y = TugShield.sensorRead();
 response(k, :) = [u, y];
 k = k + 1;
 nextStep = 0;
end

if (toc >= Ts * k)
 if (toc >= Ts * (k + 1))
 disp('Sampling violation has occurred.')
 samplingViolation = 1
 TugShield.actuatorWrite(0);
 break
 end
 nextStep = 1;
end
end

```

## Simulink

V prostredí Simulink sa navrhla základná štruktúra príkladu Identification, so zabudovanou kalibráciou pre predefinovanú trajektóriu vstupov.



Obr. 5.6: Návrh štruktúry príkladu Identification v prostredí Simulink.

### 5.2.4 Príklad s použitím PID regulátora

V praxi nestačí signál len sledovať, ale tak isto aj regulovať, aby sa dosiahli požadované hodnoty. Jeden z najjednoduchších regulátorov je práve PID regulátor, ktorý nie je viazaný s modelom sústavy, ale pracuje čistlo len so vstupom a výstupom. Pre jeho jednoduchosť a finančnú nenáročnosť je doteraz najvyužívanejším regulátorom.

Nasledovný príklad aplikuje PID regulátor na sústavu TugShield a výstupom je zregulovaný signál z flexi snímača podľa nastavenia uhla servomotora.

## Arduino IDE

Regulátor pracuje s tromi konštantami - proporcionálnou KP, integračnou KI a derivačnou KD, ktoré bolo treba na začiatku definovať. Hodnoty sa mohli určiť dvoma spôsobmi. Buď metódou pokusov a opráv alebo s pomocou MATLABu.

Základ programu je veľmi podobný s príkladom Identification s tým, že po odčítaní hodnoty z flexi snímača sa vypočíta odchýlka medzi žiadanou a skutočnou hodnotou a následne sa podľa toho zmení akčný zásah, ktorý je vstupom do systému.

V príklade sa využíva vzorkovanie v presne stanovených časových okamihov a trajektória preddefinovaných vstupov, na ktoré sa regulátor snaží uregulovať výstup.

```
#define KP 0.5
PIDAbs.setKp(KP);
void loop() {
 if (nextStep) {
 step();
 nextStep=false; } }
void stepEnable(){
 nextStep=true;}
void step(){
if (k % (T*i) == 0){
 r = R[i]+40;
 i++;}
u = PIDAbs.compute(r-y,0,180,0,100);
TugShield.actuatorWrite((int)u);
y = TugShield.sensorRead();
```

## MATLAB

Nasledujúci kód zobrazuje aplikáciu PID regulátora na TugShield v prostredí MATLAB. Hlavná časť programu sa nachádza v nekonečnom while cykle, ktorý každým krokom prepočítava a zapisuje akčné zásahy do servomotora, na základe čoho sa reguluje ohnutie nosníka.

```
tic
while (1)
 if (nextStep)
 if (mod(k, T*i) == 1)
 i = i + 1;
 if (i > length(R))
 TugShield.actuatorWrite(0.0);
 break
 end
 r = R(i);
 end
 y = TugShield.sensorRead();
 u = PID.compute(r-y, 0, 100, 0, 100);
 TugShield.actuatorWrite(u);
```

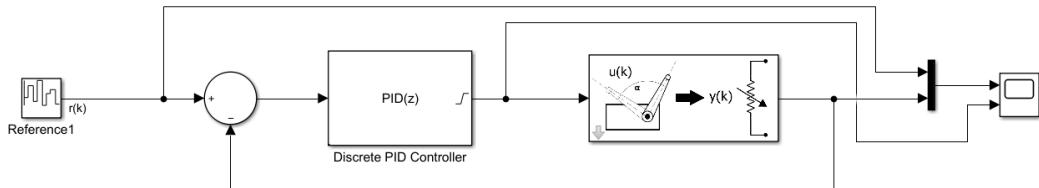
```

 response(k, :) = [r, y, u];
 k = k + 1;
 nextStep = 0;
 end
 if (toc >= Ts * k)
 if (toc >= Ts * (k + 1))
 disp('Sampling violation has occurred.')
 samplingViolation = 1
 TugShield.actuatorWrite(0);
 break
 end
 nextStep = 1;
 end
end

```

## Simulink

Aj v Simulinku sa navrhlo príklad s využitím PID riadenia. Bol však skonštruovaný ako koncept bez otestovania.



Obr. 5.7: Návrh štruktúry príkladu PID v prostredí Simulink.

## 6 Záver

Cieľom diplomovej práce bolo navrhnúť a spojazdniť miniatúrny prístroj na riadenie statickej deformácie nosníka s názvom TugShield. Shield sa skladá z dvoch hlavných komponentoch, zo servomotora a flexi senzoru. Tieto dva prvky sú navzájom prepojené a rameno servomotora úmerne ohýna nosník, čím vzniká spätno-väzobná akcia.

Práca nadväzovala na bakalársku prácu, v ktorej bol navrhnutý prvý prototyp a vytvorený softvérový základ v Arduino IDE. Samozrejme, bolo tam niekoľko rôznych nedostatkov, popísaných v druhej kapitole, ktoré vytvárali priestor pre ďalšiu prácu.

Ako prvá, sa zmenila celá logika zapojenia operačného zosilňovača. Vymenil sa aj samotný opamp, z predošlého na rail-to-rail a s napájaním 3,3V. Vytvorilo sa najprv diferenčné zapojenie, ktoré otvorilo otázku vzniku impedancií v obvode a ich riešenia. Situácia, kedy v dvoch bodoch vznikali deličky napäťia, sa riešila buffer opampom a preto sa musel do obvodu použiť TLC2274, ktorý obsahoval až štyri operačné zosilňovače.

V konečnom dôsledku vznikla veľmi komplikovaná elektrická schéma, čo bol podnet pre vytvorenie ďalšieho návrhu elektrického zapojenia, ktorý vychádzal zo základnej logiky fungovania operačných zosilňovačov. Celý postup navrhovania bol opísaný v tretej kapitole, ktorej výsledkom boli dva finálne prototypy.

Po otestovaní sa zistilo, že prototyp R2A neboli vobec funkčný a snímač nereagoval na ohyb nosníka. Zistilo sa to príkladom FirstCheck, kde snímač nevykazoval žiadnu odozvu. Prototyp R2B bol funkčný, ale zosilnenie nie je ideálne. Rozsah od minimálneho po maximálny ohyb nosníka bol reprezentovaný len 40 ADC úrovňami.

Návrh zosilnenia sa najskôr simuloval v počítačovom prostredí, ktoré sa presunulo na breadboard a následne na PCB dosku. V každom prípade sa rezistory museli nastaviť na iné hodnoty, aby bolo zosilnenie ideálne. Ako vysvetlenie je pravdepodobné, že obvod je citlivý na odpor ciest a káblu. Preto na konečnom výsledku nebolo zosilnenie ideálne a nájdenie vhodnej kombinácií rezistorov by si vyžadovalo veľmi veľa času.

V ďalších častiach sa práca venovala softvérovej časti, riadeniu a návrhu modelu. Vytvorili sa API pre tri rôzne prostredia a to Arduino IDE, MATLAB a Simulink. Základom bola knižnica TugShield, ktorá obsahovala metódy, funkcie alebo bloky, ktorými sa dali ovládať základné funkcie TugShieldu:

- TugShield.begin - inicializácia hardvéru,
- TugShield.calibration - kalibrácia TugShieldu,
- TugShield.sensorRead - čítanie signálu z flexi snímača,
- TugShield.actuatorWrite - nastavovanie ramena servomotora.

Knižnica sa testovala v demonštračných príkladoch, ktoré jednak overovali funkčnosť metód, ale aj predstavovali užívateľovi celkové fungovanie TugShieldu a jeho použitie. V každom prostredí sa vyhotovili príklady s rovnakým cieľom, aby sa medzi prostrediami dali porovnať výsledky.

Prvý príklad SelfTest bol zameraný na otestovanie kalibrácie. Sledovalo sa či hodnoty v maximálnych polohách splňajú limit. Druhý príklad Identification, na trajektórii predefinovaných vstupov zaznamenával správanie sa výstupného signálu. Tretí príklad implementoval PID riadenie na systém.

V rámci príkladov sa riešila teoretická aj experimentálna analýza systému, kde sa vytvoril koncept na vytvorenie modelu TugShieldu. Spravilo sa meranie, ktoré pomohlo charakterizovať oblasti priebehu, ktoré by mohli byť linearizované a vytvoriť pre ne jednoduchý model.

Časť príkladov bola úspešne otestovaná a funkčná, ale čo nebolo spojazdnené, bolo PID riadenie, ktoré sa nepodarilo rozbehnuť v žiadnom z prostredí. Keďže príklad Identification bol úspešný, dá sa to považovať ako chyba programu, ktorá nebola zistená. To isté je aj prípad príkladov v Simulinku, ktoré neboli vyladené tak, aby mohli byť otestované, a preto sa vytvoril len koncept týchto príkladov.

Prácu dosť ovplyvnila situácia a prístup do laboratória, ktorý neboli možný. Provizórne laboratórium, ktoré sa zriadilo doma, nebolo dostačujúce na rýchlu a efektívnu prácu.

## 6.1 Priestor na zlepšenie

Aj druhá verzia TugShieldu otvára priestor na zlepšenie. Ako prvý by sa v budúcnosti mohol plne vyladiť hardvér. Síce vznikli dve verzie R2A a R2B, ale z toho len druhá verzia R2B bola funkčná, ale zosilnenie nebolo ani z d'aleka ideálne.

Určite by sa z hľadiska hardvéru dalo zlepšiť aj upínanie serva a nosníka, ktoré je momentálne len prvoplánové. Môže sa navrhnúť moderný dizajn a minimalizovať materiál.

Softvérová časť je asi najslabšia stránka TugShieldu R2, navrhlo sa len PID riadenie, ktoré patrí medzi najjednoduchšie a v konečnom dôsledku ani tento návrh neboli funkčný. Vytvorené metódy a funkcie či bloky, by sa dali využiť v množstve kreatívnych príkladov. V tejto práci boli predstavené len tie základné a veľa z nich pre nedostatok času neboli otestované.

Čo sa týka identifikácie, bol načrtnutý len koncept na vytváranie matematicko-fyzikálneho modelu, ktorý by sa dal rozvinúť a aj využiť v budúcnosti v LQ alebo MPC riadení.

Momentálne sa na TugShielde meria ohyb reprezentovaný odporom, v nasledujúcej práci by bolo zaujímavé merať aj odchýlku nosníka od východnej pozície a dať do porovnania s uhлом.

# Literatúra

- [1] Ing. Alexander Žatkovič. OPERAČNÉ ZOSILŇOVACÉ. Online, 2018. [cit.14.3.2021], <https://alzat.spseke.sk/zosil/Jednosmr/oz/oz.htm>.
- [2] Analog Devices. LTspice. Online, 2021. [cit.27.3.2021], <https://www.analog.com/en/design-center/design-tools-and-calculators/ltpice-simulator.html>.
- [3] Andrea Chalupová. Čo očakávajú zamestnávatelia od absolventov. Online, 2020. [cit.26.7.2020], <https://hnonline.sk/prakticke-hn/188598-co-ocakavaju-zamestnavatelia-od-absolventov>.
- [4] S. Angalaeswari, A. Kumar, D. Kumar, and S. Bhadriya. Speed control of permanent magnet (PM)DC motor using Arduino and LabVIEW. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–6, 2016.
- [5] Arduino. Arduino Uno Rev3. Online, 2020. [cit.26.7.2020], <https://store.arduino.cc/arduino-uno-rev3>.
- [6] K. R. Asha, P. S. Tasleem, A. V. Ravi Kumar, S. M. Swamy, and K. R. Rekha. Real Time Speed Control of a DC Motor by Temperature Variation Using LabVIEW and Arduino. In *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, pages 72–75, 2017.
- [7] B. Bc. Jakub Kulhánek. *Embedded predictive control of a laboratory inverted pendulum system*. PhD thesis, Slovak university of technology, Faculty of mechanical engineering, Slovak university of technology, Máj 2019.
- [8] I. Cacciari, A. A. Mencaglia, and S. Siano. Smart laser ablation: An Arduino-based feedback for hand held laser delivery system. In *2015 XVIII AISEM Annual Conference*, pages 1–4, 2015.
- [9] D. P. Desai and D. M. Patel. Design of Control unit for CNC machine tool using Arduino based embedded system. In *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pages 443–448, 2015.
- [10] Digilent. What is the Analog Discovery 2? Online, 2021. [cit.3.4.2021], <https://store.digilentinc.com/analog-discovery-2-100msps-usb-oscilloscope-logic-analyzer-and-variable-power-supply/>.

- [11] Diptrace. Diptrace Software. Online, 2021. [cit.14.3.2021], <https://diptrace.com/diptrace-software/>.
- [12] W. J. Esposito, F. A. Mujica, D. G. Garcia, and G. T. A. Kovacs. The Lab-In-A-Box project: An Arduino compatible signals and electronics teaching system. In *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, pages 301–306, Aug 2015.
- [13] A. A. Galadima. Arduino as a learning tool. In *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, pages 1–4, Sep. 2014.
- [14] Gergely Takacs. AutomationShield. Online, 2020. [cit.26.7.2020], <https://github.com/gergelytakacs/AutomationShield/wiki>.
- [15] J. Hurtuk, M. Chovanec, and N. Ádam. The Arduino platform connected to education process. In *2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES)*, pages 000071–000076, Oct 2017.
- [16] D. Iosifidis, N. Alvanos, C. Yfoulis, S. Papadopoulou, and C. Galatsopoulos. Practical PID Hovering Control of a Laboratory Quadcopter Using Low-Cost Embedded Control Hardware and Software. In *2018 UKACC 12th International Conference on Control (CONTROL)*, pages 428–433, 2018.
- [17] Janet Heath. Amplifiers: What do rail-to-rail and single supply mean? Online, 2017. [cit.2.4.2021], <https://www.analogictips.com/amplifiers-rail-to-rail-single-supply-mean/>.
- [18] B. Korunur Engiz and R. Bashir. Implementation of a Speed Control System Using Arduino. In *2019 6th International Conference on Electrical and Electronics Engineering (ICEEE)*, pages 294–297, 2019.
- [19] J. Li. Control System Laboratory with Arduino. In *2018 International Symposium on Computer, Consumer and Control (IS3C)*, pages 181–184, 2018.
- [20] Martin Gulan, Gergely Takács. FloatShield. Online, 2020. [cit.26.7.2020], <https://github.com/gergelytakacs/AutomationShield/wiki/FloatShield>.
- [21] Martin Gulan, Gergely Takács. HeatShield. Online, 2020. [cit.26.7.2020], <https://github.com/gergelytakacs/AutomationShield/wiki/HeatShield>.
- [22] Martin Gulan, Gergely Takács. MagnetoShield. Online, 2020. [cit.26.7.2020], <https://github.com/gergelytakacs/AutomationShield/wiki/MagnetoShield>.
- [23] Martin Gulan, Gergely Takács. OptoShield. Online, 2020. [cit.26.7.2020], <https://github.com/gergelytakacs/AutomationShield/wiki/OptoShield>.
- [24] Massachusetts Institute of Technology. Introduction to Electrical Engineering and Computer Science I. Online, 2011. [cit.21.4.2021], <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/>.

- [25] Mathworks. System Identification Toolbox. Online, 2021. [cit.10.5.2021], <https://www.mathworks.com/help/ident/>.
- [26] B. M. Matušík. *Modelovanie a riadenie elektronickej škrtiacej klapky s využitím po častiach a finného modelu*. PhD thesis, Slovenská technická univerzita, Strojnícka fakulta, Slovenská technická univerzita, Máj 2017.
- [27] MechaniCalc. Beam Deflection Tables. Online, 2014. [cit.19.5.2021], <https://mechanicalc.com/reference/beam-deflection-tables>.
- [28] M. Novák, J. Kalová, and J. Pech. Use of the Arduino Platform in Teaching Programming. In *2018 IV International Conference on Information Technologies in Engineering Education (Inforino)*, pages 1–4, Oct 2018.
- [29] O. K. Ogidan, K. O. Temikotan, and K. C. Chike. Development of an Arduino microcontroller-based automatic load shedding module for teaching and research. In *2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON)*, pages 1130–1134, Nov 2017.
- [30] C. A. Petry, F. S. Pacheco, D. Lohmann, G. A. Correa, and P. Moura. Project teaching beyond Physics: Integrating Arduino to the laboratory. In *2016 Technologies Applied to Electronics Teaching (TAEE)*, pages 1–6, June 2016.
- [31] Profesia. Kto boduje na trhu prace najviac. Online, 2020. [cit.26.7.2020], <https://firma.profesia.sk/kto-boduje-na-trhu-prace-najviac-toto-jerebricek-najziadanejsich-absolventov-vysokych-skol/>.
- [32] Sario. Automobilový priemysel. Online, 2020. [cit.26.7.2020], <https://www.sario.sk/sk/investujte-na-slovensku/sektorove-analyzy/automobilovy-priemysel>.
- [33] J. Sheng. Real Time DC Water Tank Level Control using Arduino Mega 2560. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 635–640, 2019.
- [34] Sparkfun. Flex Sensor 2.2. Online, 2016. [cit.14.3.2021], <https://www.sparkfun.com/products/10264>.
- [35] SparkFun. Getting Started with LTspice. Online, 2021. [cit.2.4.2021], <https://learn.sparkfun.com/tutorials/getting-started-with-ltspice/all>.
- [36] T. Teslyuk, P. Denysyuk, A. Kernytskyy, and V. Teslyuk. Automated control system for Arduino and android based intelligent greenhouse. In *2015 XI International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pages 7–10, 2015.
- [37] Tibor Konkoly, Gergely Takács. MotoShield. Online, 2020. [cit.26.7.2020], <https://github.com/gergelytakacs/AutomationShield/wiki/MotoShield>.

- [38] B. E. Vargová. *Experimentálne zariadenie na riadenie statickej deformácie votknutého nosníka*. PhD thesis, Slovenská technická univerzita, Strojnícka fakulta, Bratislava, Slovenská Republika, Máj 2019.
- [39] B. F. Čelko. *Experimentálne zariadenie na praktickú implementáciu adaptívneho prediktívneho riadenia*. PhD thesis, Slovenská technická univerzita, Strojnícka fakulta, Slovenská technická univerzita, Máj 2018.
- [40] P. V. Vimal and K. S. Shivaprakasha. IOT based greenhouse environment monitoring and controlling system using Arduino platform. In *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 1514–1519, 2017.
- [41] Vladimír Amrich. Čo spôsobí koronakríza na slovenskom pracovnom trhu? Online, 2020. [cit.26.7.2020], <https://finweb.hnonline.sk/ekonomika/2154712-co-sposobi-koronakriza-na-slovenskom-pracovnom-trhu>.
- [42] Wikipedia. Deflection (engineering). Online, 2021. [cit.19.5.2021], [https://en.wikipedia.org/wiki/Deflection\\_\(engineering\)](https://en.wikipedia.org/wiki/Deflection_(engineering)).
- [43] WISC. What Is MATLAB? Online, 2021. [cit.27.5.2021], <https://cimss.ssec.wisc.edu/wxwise/class-aos340/spr00/whatismatlab.htm>.